WebSphere® Process Server

IBM

Version 6.0

Monitoring

Note

Before using this information, be sure to read the general information in “Notices” on page 43.

September 29 2005

This edition applies to version 6, release 0, of WebSphere Process Server (product number 5724-L01) and to all subsequent releases and modifications until otherwise indicated in new editions.

To send us your comments about this document, email doc-comments@us.ibm.com. We look forward to hearing from you.

Contents

Monitoring

This section describes how to monitor service components in IBM® WebSphere® Process Server, Version 6.0.

WebSphere Process Server documentation PDFs.

Service component monitoring overview

This section will give you a conceptual overview of the reasons you would monitor service components on the process server; which event points within the service components you would select to monitor; and, how to configure monitoring on your system.

WebSphere Process Server provides capabilities for monitoring service components to aid in system administration functions, such as performance tuning and problem determination. It goes beyond these traditional functions by also providing the capability for persons who are not necessarily information technology specialists to continually monitor the processing of the service components within the applications deployed on your system. By overseeing the overall processing flow of the interconnected components, you can ensure that your system is producing what you expect it to produce.

WebSphere Process Server operates on top of an installation of WebSphere Application Server, and, consequently, uses much of the functionality of the application server infrastructure for monitoring system performance and troubleshooting. It also includes some extra functionality that is specifically designed for monitoring process server service components. This section of the WebSphere Process Server Information Center will focus on how you would monitor process server-specific service components. It is intended to supplement the monitoring and troubleshooting topics found in the WebSphere Application Server Version 6.0 Information Center. Refer to the application server documentation for details of the other monitoring capabilities in the combined product.

Why use monitoring?

This section presents the various reasons you would monitor service components within WebSphere Process Server: to assess performance; to troubleshoot problems; and, evaluate the overall processing of service components that make up the applications deployed on your system.

Service components are the integral functions incorporated into WebSphere Process Server, that allow you to easily create and deploy applications on your system that mirror the processes employed in your enterprise. Effectively monitoring those service components is, therefore, essential to managing the tasks which the process server is intended to accomplish. The main reasons you would monitor service components on the process server are outlined below:

Problem determination
: You can diagnose particular errors by using the logging and tracing facilities provided by WebSphere Application Server, which underlies WebSphere Process Server. For example, if a particular application is not

producing the expected results, you can set up a logger to monitor the processing of the service components that comprise that application. You can have the log output published to a file, which you can then examine to pinpoint the cause of the problem. Troubleshooting is a task that is of importance to system administrators and others concerned with the maintenance of system hardware and software.

Performance tuning
You can monitor certain performance statistics that most process server-specific service components produce. Use this information to maintain and tune your system health, and ensure that your applications are tuned optimally and efficiently. You can also spot situations where one or more of your services are performing at a poor level, which may indicate that other problems are present in your system. Like problem determination, performance tuning is a task typically performed by information technology specialists.

Assessing the processing of service components
Problem determination and performance tuning are tasks you would be done on a short-term basis, to solve a particular issue or problem. You can also set up the process server to continually monitor the service components incorporated into the applications deployed on your system. This type of service component monitoring is of importance to those who are responsible for designing, implementing, and ensuring that the processes achieve their design goals, and may be accomplished persons who are not necessarily specialists in information technology.

What do you monitor?

Service component monitoring is enabled on WebSphere Process Server by selecting certain point that service component reaches when it is processed. This section explains how each service component defines these points, and the results of an invocation of a monitor on a service component when it detects some activity on a specific point.

Regardless of the type of monitoring you intend to perform on your service components (performance, tuning, or process monitoring), you will actually be monitoring a certain point that is reached during the processing of a service component. This point is referred to as an *event point*, and it is these points that you would select to be monitored. Each event point encapsulates the processed *element* of a service component, and the *nature* of the event. Both of these factors will determine the type of event fired by monitoring.

A service component type will support monitoring for one or more element kinds, each with its own set of event natures. Different element kinds within a component indicate that some functions of the component are processed under certain circumstances, and the others are processed under other circumstances. Event natures are key positions that are reached when a service component element kind is processed. The most common natures for service component events are ENTRY, EXIT, and FAILURE, but there are many other natures depending on the particular component and element.

An example of how a service component type may contain different element kinds, each possessing its own set of event natures, would be the map component, which supports two element kinds: one for the main map function (named WBI.MAP), and a second for processing steps within the main function (named WBI.MAP.Transformation). Both elements have natures of ENTRY, EXIT, and

FAILURE, resulting in a total of six possible event types that can be monitored for the entire map component. The list of all service component elements and their event natures is contained in the event catalog.

Monitoring is a separate layer of functionality that lies atop the processing of your applications, and does not interfere with the processing of your service components. Monitoring is concerned with service component processing only insofar as it detects activity at a specified event point. When this happens, an event is fired by monitoring, which determines where the event is sent, and what data is contained in that event, based on the type of monitoring you are performing, as detailed below:

Performance metrics

If you are monitoring a service component in order to gather performance metrics, light weight events are fired to the Performance Monitoring Infrastructure. You can select for monitoring one or more of the three performance statistics generated for process server-specific server components:

- A counter for each EXIT event nature – this counts successful computations.
- A counter for each FAILURE event nature – this counts failed computations
- The processing duration calculated between corresponding ENTRY and EXIT events (synchronous computations only).

Common Base Events

If you wish to capture the data from events fired by monitoring at specified event points in service component, then you would configure the process server to generate the event and its data to be encoded in Common Base Event formats. Common Base Events capture the business data passed on invocations, responses, and possible failure points of any service request. You can choose to publish these events to either a logger or to the Common Event Infrastructure (CEI) bus, which directs the output to a specially configured CEI server database.

How do you enable monitoring?

This section gives a brief overview of the different ways that you can specify service component event points for monitoring, depending on the type of monitoring you are planning to do on the process server.

Performance statistics

Use the Performance Monitoring Infrastructure (PMI) section of the administrative console to specify the particular event points and their associated performance measurements that you wish to monitor. After you start monitoring service component performance, the generated statistics are published at certain intervals to the Tivoli® Performance Viewer. You can use this viewer use to watch the results as they occur on your system, and, optionally, log the results to a file that can be later viewed and analyzed within the same viewer.

Common Base Events for problem determination and business process monitoring

You can specify, at the time you create an application, to monitor service component event points — along with a certain level of detail for those events — on a continual basis after the application is deployed on a running server. You can also select event points to monitor after the application has been deployed and the events invoked at least once on the process server. In both cases, the events generated by monitoring will be fired across the Common Event Infrastructure (CEI) bus. These events can be published to a log file, or to a configured CEI Server database. WebSphere Process Server supports two types of Common Base Event enablement for problem determination and business process monitoring:

Static Certain events points within an application and their level of detail can be tagged for monitoring using the WebSphere Integration Developer tooling. The selections indicate what event points are to be continuously monitored, and are stored in a file with a .mon extension this is distributed and deployed along with the process server application. Once the process server is configured to use a CEI server, the monitoring function will begin firing Common Base Events to a CEI server whenever the specified services are invoked. As long as the application is deployed on the process server, the service component event points specified in the .mon file will be constantly monitored until the application is stopped. You can specify additional events to be monitored in a running application, and increase the detail level for event points that are already monitored. But as long as that application remains active you cannot stop, or lower the detail level of, the monitored event points specified by the .mon of the deployed application.

Dynamic

If additional event points need to be monitored during the processing of an application without shutting down the server, then you can use dynamic monitoring. Use the administrative console to specify service component event points for monitoring, and set detail level for the payload that will be included in the Common Base Event. A list is compiled of the event points that have been reached by a processed service component after the process server was started. Choose from this list individual event points or groups of event points for monitoring, with the Common Base Events directed either to the logger or to the CEI server database.

The primary purpose of the Dynamic enablement is for creating correlated Common Base Events that are published to logs, which allow you to perform problem determination on services. While you can also send additional events to the CEI server, this may not be the best way to resolve problems. Common Base Events can be large, depending on how much data is being requested, and so care should be taken to not permanently affect the performance of the primary activities on your server.

Enabling and configuring service component monitoring

In order to monitor the service components on the process server, you must first enable the monitoring capabilities. Then you must specify the events you wish to monitor, the information you wish to capture from the event, and the method used to publish the results. This section contains details on how you would accomplish these tasks.

Monitoring performance

Performance measurements are available for service component event points, and are processed through the Performance Monitoring Infrastructure. This section describes how you would configure the process server to gather performance metrics from service component event points.

Whether you are tuning WebSphere Process Server service components for optimal efficiency or diagnosing a poor performance, it is important to understand how the various run time and application resources are behaving from a performance perspective. The Performance Monitoring Infrastructure (PMI) provides a comprehensive set of data that explains the runtime and application resource behavior. Using PMI data, the performance bottlenecks in the application server can be identified and fixed. PMI data can also be used to monitor the health of the application server.

The PMI is included in the base WebSphere Application Server installation, and this functionality is documented in considerable detail in the WebSphere Application Server Version 6.0 Information Center. This section provides only supplemental information about performance monitoring as it relates to the service components specific to WebSphere Process Server, so consult the information in the WebSphere Application Server documentation for using PMI with other parts of the entire product.

The service component event points specific to WebSphere Process Server that can be monitored by the PMI are those that typically have ENTRY, EXIT, and FAILURE event natures. Event sources which are not defined according to this pattern are not supported; this includes business process or human task events. Events that are supported have three types of performance statistics that can be measured:

- Successful invocations.
- Failed invocations.
- Elapsed time for event completion.

Enabling PMI using the administrative console

To monitor performance data you must first enable the Performance Monitoring Infrastructure on the server.

You can enable the Performance Monitoring Infrastructure (PMI) through the administrative console.

1. Open the administrative console.
2. Click **Servers** > **Application Servers** in the console navigation tree.
3. Click *server*.
4. Click the **Configuration** tab.
5. Click **Performance Monitoring Infrastructure** under Performance.
6. Select the **Enable Performance Monitoring Infrastructure (PMI)** check box.
7. **Optional:** Select the check box for **Use sequential update** to enable precise statistic update.
8. Go back to the server PMI configuration page by clicking the server name link.
9. Click **Apply** or **OK**.
10. Click **Save**.
11. Restart the process server.

The changes you make will not take effect until you restart the process server.

Specifying service components to monitor with PMI

You can specify single events, multiple events, or groups of related events for monitoring through the Performance Monitoring Infrastructure by using the administrative console. You would also use the administrative console to specify which performance statistics you wish to capture for each event.

Ensure that you have enabled performance monitoring, and that you have at least once invoked the event you wish to monitor before performing this task.

1. Open the administrative console.
2. Click **Servers** > **Application Servers** in the console navigation tree.
3. Click *server*.
4. Click the **Runtime** tab.
5. Click **Performance Monitoring Infrastructure** under Performance.
6. Select **Custom**.
7. Expand **WBIStats.RootGroup** . All of the process server-specific events that can be monitored are within this group. Some events may not be listed because they have not been invoked since the process server was last started.
8. Choose an event you wish to monitor from within the tree on the left-hand side of the panel, and then select the statistics that you wish to collect on the right-hand side and click **Enable**. Repeat this for all events that you wish to monitor.
9. Go back to the server PMI configuration page by clicking the server name link.
10. Click **Apply** or **OK**.
11. Click **Save**.

You can now start monitoring the performance of your chosen events in the Tivoli Performance Viewer.

Monitoring Common Base Events

WebSphere Process Server monitoring can capture the data in a service component at a certain event point. These events are formatted in a standard called the Common Base Event. You can have the process server publish these events to the logging facilities, or you can utilize the more versatile monitoring capabilities of a Common Event Infrastructure server. This section details how you would perform these tasks.

Enabling service component monitoring

You must configure WebSphere Process Server to support monitoring service components before you do any actual monitoring.

You must have previously created the business process container and the human task container on the process server.

Perform this task to enable Common Event Infrastructure (CEI) monitoring support on WebSphere Process Server.

1. Open the administrative console.
2. In the left frame, expand **Servers** and click the **Application servers** link.
3. In the right frame, click your server name link.
4. Complete the following steps to enable the server to generate business process events onto the CEI server:

a. Under the Containers Settings section, expand **Business process container settings**.
b. Click **Business process container**.
c. Select the **Enable Common Event Infrastructure logging** check box.
d. Select the **Enable audit logging** check box and click **OK**.
e. Click the **Apply** button.
f. Click the **Save** link and then click the **Save** button.
g. Click the server name link to return to the properties for the server.

5. Complete the following steps to enable the server to generate human task events onto the CEI server:
   a. Under the Containers Settings section, expand **Human task container settings**.
   b. Click on **Human task container**.
   c. Select the **Enable Common Event Infrastructure logging** check box.
   d. Select the **Enable audit logging** check box and click **OK**.
   e. Click the **Apply** button.
   f. Click the **Save** link and then click the **Save** button.
   g. Click the **Logout** button and then close the administrative console.

If the process server was already started, then you must restart it for the changes to take effect.

Logging Common Base Events

You may choose to use the logging facilities of WebSphere Application Server to capture the Common Base Events fired by process server monitoring. Use the loggers to view the data in events when you diagnose problems with the processing of your applications.

WebSphere Process Server uses the extensive logging facilities of the underlying WebSphere Application Server to allow you to capture the events fired by process server monitoring at service component event points. You can use the administrative console to specify the particular service component event points that you wish to monitor, the amount of payload detail contained in the resulting Common Base Events, and the method used to publish the results — such as to a file of a certain format, or directly to a console. Monitor logs contain events encoded in Common Base Event format, and you can use the information contained in the event elements to trace problems with the processing of your service components.

The functionality of WebSphere Application Server logging and tracing capabilities is documented in considerable detail in the WebSphere Application Server Version 6.0 Information Center, with complete details of how logging and tracing is used within the entire product. This section provides only supplemental information about logging as it relates to the service components that are specific to WebSphere Process Server. Consult the information in the WebSphere Application Server documentation for using logging and trace with other components of the entire product.

Enabling the diagnostic trace service:

Use this task to enable the diagnostic trace service, which is the logging service that can manage the amount of detail contained in the Common Base Event.

You must have the business process and human task containers configured to allow Common Event Infrastructure (CEI) logging and audit logging.

The diagnostic trace service is the only logger type that can provide the level of detail required to capture the detail contained in the elements of Common Base Events. You must enable the diagnostic trace service before you start the process server in order to log events. The service must also be enabled if you use the administrative console to select service component event points for monitoring using the CEI server.

1. In the navigation pane, click **Servers > Application Servers**.
2. Click the name of the server that you want to work with.
3. Under Troubleshooting, click **Diagnostic Trace service**.
4. Check **Enable log** on the **Configuration** tab.
5. Click **Apply**, and then **Save**.
6. Click **OK**.

If the process server was already started, then you must restart it for the changes to take effect.

Configuring logging properties using the administrative console:

Use this task to specify that the monitoring function publish Common Base Events to a logger.

Before WebSphere Process Server applications can log monitored events, you must specify the service component event points that you wish to monitor, what level of detail you require for each event, and format of the output used to publish the events to the logs. Using the administrative console, you can:

- Enable or disable a particular event log.
- Specify the level of detail in a log.
- Specify where log files are stored, how many log files are kept, and a format for log output.

You can change the log configuration statically or dynamically. Static configuration changes affect applications when you start or restart the application server. Dynamic or run time configuration changes apply immediately.

When a log is created, the level value for that log is set from the configuration data. If no configuration data is available for a particular log name, the level for that log is obtained from the parent of the log. If no configuration data exists for the parent log, the parent of that log is checked, and so on up the tree, until a log with a non-null level value is found. When you change the level of a log, the change is propagated to the children of the log, which recursively propagates the change to their children, as necessary.

1. Enable logging and set the output properties for a log:
2. In the navigation pane, click **Servers > Application Servers**.
3. Click the name of the server that you want to work with.
4. Under Troubleshooting, click **Logging and tracing**.
5. Click **Change Log Detail levels**.
6. The list of components, packages, and groups displays all the components that are currently registered on the running server; only process server events that have been invoked at least once will appear on this list. All process server

components with event points that can be logged are listed under one of the components that start with the name **WBILocationMonitor.LOG.**

- To select events for a static change to the configuration, click the **Configuration** tab.
- To select events for a dynamic change to the configuration, click the **Runtime** tab.

7. Select the event or group of events that you wish to log.
8. Set the logging level for each event or group of events.

   Note: Only the levels FINE, FINER, and FINEST are valid for CEI event logging.
9. Click **Apply**.
10. Click **OK**.
11. To have static configuration changes take effect, stop then restart the process server.

By default, the loggers will publish their output to a file called trace.log, located in the <*install_root*>/profiles/<*profile_name*>/logs/<*server_name*> folder.

Monitoring service components with the CEI server

You can choose to have service component monitoring results published to a Common Event Infrastructure server. These events are structured identically to the events sent to loggers, but are stored on a database which can be accessed by viewers designed specifically for analyzing Common Base Events. Service component event points can be specified for monitoring with the Common Event Infrastructure server on a permanent basis for viewing and managing application flow, or on an ad-hoc basis for troubleshooting problems.

A unique capability of WebSphere Process Server monitoring is its ability to publish the data in service component event points within Common Base Events that are fired across the Common Event Infrastructure (CEI) bus. This approach to monitoring allows you much more flexibility in analyzing your service component activities on your system. You can also utilize browsers optimized for CEI events — such as the Common Base Event browser, which is included with the process server.

Service component event points can be specified within an application, when it is created, for continual monitoring at all times after the application is deployed and running on a server — a method known as ″static″ monitoring. You would perform static monitoring on service component event points that are of particular importance in the proper flow of component processing on your system. With this information, you can easily oversee the overall actions of, and interactions between, the service component processes running on your system. You would also have the ability to quickly detect deviations from the normal flow of these processes, which may indicate that your service components are not working properly.

To configure static monitoring of service components, you would use WebSphere Integration Developer to select the service component event points in your applications that will be deployed on the process server. The selections are specified in the form of an XML file with a .mon extension that will be deployed along with the application. Once deployed on a running server, you will not be able to turn off or lower the detail level of the monitoring for events specified in the .mon file of the application; you must stop the server and undeploy the application to stop this kind of monitoring. Consult the WebSphere Integration Developer Information Center for details on creating and deploying applications with .mon files.

You can also select service component event points for ″dynamic″ monitoring, which can be enabled and disabled on an application already deployed a running server. The rationale for performing dynamic monitoring using the CEI server is essentially the same as that for logging: to diagnose and troubleshoot problems on your system. The output is essentially the same as that which is published to loggers, with Common Base Event elements comprising the structure for each event fired across the CEI bus. Also, like logging data, the differences in detail levels affect only how much of the payload is encoded within the event.

Configuring CEI server monitoring using the administrative console:

Use the administrative console to dynamically specify the monitoring function publish Common Base Events to the Common Event Infrastructure server.

You must enable the diagnostic trace service, just as you would with the logger. After you restart your server you would invoke the events you wish to monitor once, so that they appear on the list of events available for monitoring.

This method of selecting events for monitoring is used for applications that have already been deployed on a process server. Events that are specified in a .mon file that is deployed with the application on the process server will be monitored by the Common Event Infrastructure (CEI) database regardless of any changes you make here. For those events, you can only specify a greater level of detail to be captured and published to the CEI database. The output that is published to the CEI database is very similar to that published by loggers.

1. In the navigation pane, click **Servers > Application Servers**.
2. Click the name of the server that you want to work with.
3. Under Troubleshooting, click **Logging and tracing**
4. Click **Change Log Detail levels**
5. The list of components, packages, and groups displays all the components that are currently registered on the running server; only process server events that have been invoked at least once will appear on this list. All process server events that can be logged are listed under one of the components that start with the name **WBILocationMonitor.CEI**.
   - To make a static change to the configuration, click the **Configuration** tab.
   - To change the configuration dynamically, click the **Runtime** tab.
6. Select an event or group of events to monitor.
7. Choose the level of detail that you wish to capture for each event.

   Note: Only the levels FINE, FINER, and FINEST are valid for CEI events.
8. Click **Apply**, and then **Save**.
9. Click **OK**.
10. If you made a static change to the configuration, then you will have to restart the process server for the changes to take effect.

You can view the monitored event results in the Common Base Event browser.

Viewing monitored events

There are a number of ways for you to view the published results of your monitored events, depending on the type of monitoring your using. This section presents methods that you would to for viewing performance data, event logs, and Common Base Events stored on a Common Event Infrastructure database

Viewing performance metrics with the Tivoli Performance Viewer

This topic explains how you can use the Tivoli Performance Viewer to start and stop performance monitoring; view Performance Monitoring Infrastructure data in chart or table form as it occurs on your system; and, optionally, log the data to a file that you can later review in the same viewer.

It is assumed that one or more servers have been created and are running on the node, that the Performance Monitoring Infrastructure (PMI) is enabled, and that the service component event points that you wish to monitor have been invoked at least once so that they can be selected from within the viewer.

The Tivoli Performance Viewer (TPV) is a powerful application that allows you view a variety of details of all aspects of the performance of your process server. The section entitled ″Monitoring performance with Tivoli Performance Viewer″ in the WebSphere Application Server Version 6.0 Information Center contains details on how to use this tool for a variety of purposes, and you should consult this resource for complete instructions on using this program. This section will be limited to discussing the viewing of performance data for WebSphere Process Server-specific events.

The performance viewer enables administrators and programmers to monitor the current health of WebSphere Process Server. Because the collection and viewing of data occurs on the process server, performance is affected. To minimize performance impacts, monitor only those servers whose activity you want to monitor.

- **View current performance activity**
  1. Click **Monitoring and Tuning > Performance Viewer > Current Activity** in the administrative console navigation tree.
  2. Select **Server**, and click the name of the server whose activity you want to monitor. You can alternatively select the check box for the server whose activity you want to monitor, and click **Start Monitoring**. To start monitoring multiple servers at the same time, select the servers and click **Start Monitoring**.
  3. Select **Performance Modules**.
  4. Place a check mark in the check box beside the name of each performance module that you want to view. All process server-specific events that emit performance statistics, and that have been invoked at least once, are listed under the **WBIStats.RootGroup** hierarchy. Expand the tree by clicking **+** next to a node and shrink it by clicking **–** next to a node.
  5. Click on **View Modules**. A chart or table providing the requested data is displayed on the right side of the page. Charts are displayed by default.

     Each module has several counters associated with it. These counters are displayed in a table underneath the data chart or table. Selected counters are displayed in the chart or table. You can add or remove counters from the chart or table by selecting or deselecting the check box next to them. By default, the first three counters for each module are shown.

You can select up to 20 counters and display them in the TPV in the **Current Activity** mode.

6. Optional: To remove a module from a chart or table, deselect the check box next to the module and click **View Modules** again.
7. Optional: To view the data in a table, click **View Table** on the counter selection table. To toggle back to a chart, click **View Graph**.
8. Optional: To view the legend for a chart, click **Show Legend**. To hide the legend, click **Hide Legend**.
9. When you have finished monitoring the performance of your events, click on **Tivoli Performance Viewer**, select the server you were monitoring, and click **Stop Monitoring**.

- **Log performance statistics**

  While monitoring is active on a server, you can log the data from all the PMI counters that are currently enabled and record the results in a TPV log file. You can view the TPV log file for a particular time period multiple times, selecting different combinations of up to 20 counters each time. You have the flexibility to observe the relationships among different performance measures in the server during a particular period of time.

  1. Click on **Start Logging** when viewing summary reports or performance modules.
  2. When finished, click **Stop Logging**. By default, the log files are stored in the <install_root>/profiles/<profile_name>/logs/tpv directory on the node on which the server is running. The TPV automatically compresses the log file when it finishes writing to it to conserve space. At this point, there must only be a single log file in each zipped file and it must have the same name as the zipped file.
  3. Click **Monitoring and Tuning > Performance Viewer > View Logs** in the administrative console navigation tree to view the logs

Viewing and interpreting log output

This topic discusses how you would interpret the information presented in a log file. Depending on the type of output you selected for your logs, you can view the log files in the log viewer on the administrative console, the Log Analyzer program that is installed with the product, or in a separate text file editor of your choice.

Events fired to the logger by service component monitoring are encoded in Common Base Event format. When published to a log file, the event is included as a single, lengthy line of text in XML tagging format, which will also include several logger-specific fields, as outlined below. You should consult the event catalog section of this documentation for details on deciphering the Common Base Event coding of the logged event. Use this section to understand the other fields contained in each entry of the log file, and how the format you chose for the log file when you configured the logger is structured.

Basic and advanced format fields

On a process server, logging output can be directed either to a file or to an in-memory circular buffer. If trace output is directed to the in-memory circular buffer, it must be dumped to a file before it can be viewed. Output is generated as plain text in either basic, advanced or log analyzer format as specified by the user. The basic and advanced formats for output are similar to the basic and advanced formats that are available for the message logs. Basic and Advanced Formats use many of the same fields and formatting techniques. The fields that can be used in these formats include:

TimeStamp
: The timestamp is formatted using the locale of the process where it is formatted. It includes a fully qualified date (YYMMDD), 24 hour time with millisecond precision and the time zone.

ThreadId
: An 8 character hexadecimal value generated from the hash code of the thread that issued the trace event.

ThreadName
: The name of the Java™ thread that issued the message or trace event.

ShortName
: The abbreviated name of the logging component that issued the trace event. This is typically the class name for WebSphere Process Server internal components, but may be some other identifier for user applications.

LongName
: The full name of the logging component that issued the trace event. This is typically the fully qualified class name for WebSphere Process Server internal components, but may be some other identifier for user applications.

EventType
: A one character field that indicates the type of the trace event. Trace types are in lower case. Possible values include:

>	a trace entry of type method entry.
<	a trace entry of type method exit.
1	a trace entry of type fine or event.
2	a trace entry of type finer.
3	a trace entry of type finest, debug or dump.
Z	a placeholder to indicate that the trace type was not recognized.

ClassName
: The class that issued the message or trace event.

MethodName
: The method that issued the message or trace event.

Organization
: The organization that owns the application that issued the message or trace event.

Product
: The product that issued the message or trace event.

Component
: The component within the product that issued the message or trace event.

Basic format

Trace events displayed in basic format use the following format:

```
<timestamp><threadId><shortName><eventType>[className][methodName]<textmessage>
            [parameter 1]
            [parameter 2]
```

Advanced format

Trace events displayed in advanced format use the following format:

```
<timestamp><threadId><eventType><UOW><source=longName>[className][methodName]
<Organization><Product><Component>[thread=threadName]
<textMessage>[parameter 1=parameterValue][parameter 2=parameterValue]
```

Log analyzer format

Specifying the log analyzer format allows you to open trace output using the Log Analyzer tool, which is an application included with WebSphere Application Server. This is useful if you are trying to correlate traces from two different server processes, because it allows you to use the merge capability of the Log Analyzer.

Viewing events with the Common Base Event browser

Use the Common Base Event browser to select, sort, and view events.

This task assumes you are logged into the WebSphere Process Server administrative console.

The event browser uses the event access interface to query event data. The results of the query are shown in the browser.

1. Begin by opening the event browser. Select **Integration Applications** and then **Common Base Event Browser** in the navigation pane of the administrative console.
2. Specify the events you want to view.
3. Select the view of the returned events.
4. In any of the browser panels, when you have finished selecting search or sort criteria, click the **Get Events** button at the bottom of the browser panel to display the desired events.

Specifying the events to view

How to use the Common Base Event browser to specify search criteria for querying events in the event database.

This task assumes that you have already opened the event browser and are viewing the Get Events panel.

The Event Data Store Properties fields require completion. The Event Filter Properties fields are optional, and allow you to narrow your events search based on time, date, server name, sub-component name, and event severity parameters.

1. **Required:** Specify the Event Data Store to search.

   The field is a Java Naming and Directory Interface (JNDI) name, an Enterprise JavaBeans (EJB) reference that can be configured in the administrative console. The WebSphere Process Server default is `java:comp/env/eventsaccess`.
2. **Required:** Specify the Event Group to search.

   This is the event group from which events are retrieved. The default group is `All events`.
3. **Required:** Specify the number of events to retrieve.

   The maximum number of events to search is `500`.
4. **Optional:** Specify the Creation Date (calendar period) for the report.

   Enter the start and end dates.
5. **Optional:** Specify the Creation Time (time period) for the report.

   Enter the start and end times.
6. **Optional:** Specify the server name.
7. **Optional:** Specify the sub-component name, if applicable.
8. **Optional:** Specify an event's priority. The range of events priorities to retrieve is from 0 (lowest priority) to 100 (highest priority).

9. **Optional:** Specify an event's severity.

   The range of events severities to retrieve is from 0 (least severe) to 70 (most severe).
10. Click **Get Events**.

The number of Common Base Events matching the search criteria is displayed. If the results you queried do not appear, see *"Troubleshooting the Common Base Event browser"*.

To view the returned events, select a view from the navigation bar. You can choose **All Events**, **BPEL Process Events**, **User Data Events**, or **Server Events**. When you view event data, you can change your search criteria at any time by clicking **Get Events**.

Once events are returned, you can work with them to get various levels of event detail.

Working with events returned from the event browser

You use the event browser to view the events returned from a query.

This task acts on data that is returned by a submitted query, as described in *Specifying the events to view*.

The query returns all the events that meet your criteria.

1. Select a view from the navigation bar.

   The navigation bar offers the following views of the returned query:

   All Events
      All the events returned.

   BPEL Process Events
      Business Process Choreographer events for a specific process instance.

   User Data Events
      Events with the extension name `ECS:UserDataEvent`. This event type is created by the `addUserDataEvent` method of the ECSEmitter class.

   Server Events
      Events for a specific server.
2. Perform one of the following actions.
   - If you select **BPEL Process Events** in step 1, you must select a process template, and then a process instance.
   - If you select **Server Events** in step 1, you must select a server.
3. Click an event, to display the event data in the pane at the bottom of the browser window.

Troubleshooting the Common Base Event browser

There are four primary conditions under which you will be unable to access the Common Base Event browser.

Conditions

″Cannot find server″
   WebSphere Process Server (or network server) is unavailable. When you attempt to launch the event browser URI, a ″Cannot find server″ browser

page will be returned, which indicates that the server is unavailable. In this case, you need to contact the IBM Help Desk to determine the cause of the problem.

″File not found″
WebSphere Process Server is available; however, the event browser application may not be installed or started on the server. When you attempt to launch the event browser URI, a ″File not found″ browser page will be returned, which indicates that the server is available, but the URI is not available on that server. In this case, you need to contact the IBM Help Desk to determine the cause of the problem.

Logon panel appears
The WebSphere Process Server and the event browser are available; however, you have not been mapped to the proper role to allow access to the event browser. You will be prompted with a logon panel. When you enter your userID and password, attempting to log in, the login will fail. In this case, you need to contact the IBM Help Desk to get the proper authorization to launch the event browser.

Error message on ″Get event data″ panel
The WebSphere Process Server and the event browser are available, and you have the proper authority to gain access; however, the Common Event Infrastructure server is unavailable. An error message will be displayed on the event browser **Get Events** panel, when you click the **Get Events** button. The error information will be logged to the message log.

Event catalog

The event catalog contains the specifications for all the events that can be monitored for each service component type, as well as the associated Common Base Event extended data elements produced by each event. You should use the information presented in this section as reference material that will enable you understand how individual events are structured. This knowledge will help you decipher the information contained in each event, so that you can quickly identify the pieces of information you need from the relatively large amount of data generated by each event. The information included in this section covers the structure and standard elements of the Common Base Event; the list of events for the Business Process Choreographer service components and WebSphere Process Server-specific service components; and, the extensions to the Common Base Event unique to each event type. There is also a discussion of how business objects that may be processed by a service component are captured by in Common Base Events.

When an event of a given type is fired across the Common Event Infrastructure (CEI) bus to the CEI server or to a logger, it takes the form of a Common Base Event — which is, essentially, an XML encapsulation of the event elements created according to the event catalog specification. The Common Base Event includes a set of standard elements, process server component identification elements, Event Correlation Sphere identifiers, and additional elements unique to each event type. All of these elements are passed to the CEI server or logger whenever an event is fired by a service component monitor, with one exception: if the event includes the business object code within the payload, you may specify the amount of business object data that you wish to include in event.

The Common Base Event standard elements

The elements of the Common Base Event that are included in all events fired from service component monitoring are listed here. They include the Common Base Event headers, the service component identification elements, the Event Correlation Sphere identification elements, and the situation identification elements.

Attribute	Description
	CommonBaseEvent element
creationTime	The time at which the event is created, in UTC.
globalInstanceId	The identifier of the Common Base Event instance. This ID is automatically generated.
localInstanceId	This ID is automatically generated (may be blank).
sequenceNumber	This value is set by the event factory.
severity	Not used.
priority	Not used.
version	Set to `1.0.1`.
reporterComponentId	Not used.
extensionName	Set to the event name.
	sourceComponentId element
component	Process server version.
componentIdType	set to `Component Kind QName`.
executionEnvironment	A string that identifies the operating system.
instanceId	The identifier of the server. This identifier has the format *`cell name/node name/server name`*. The delimiters are operating system dependent.
location	Set to the host name of the executing server.
locationType	Set to `Hostname`.
processId	The process identifier of the operating system.
subComponent	The observable element name.
application	Not used.
threadId	The thread identifier of the Java virtual machine (JVM).
componentType	The component QName, based on the Apache QName format.
	contextDataElement element
ECSCurrentID	The value of the current Event Correlation Sphere ID.
ECSParentID	The value of the parent Event Correlation Sphere ID.
	situation element
categoryName	For process server-specific components, set to `STATUS`.
situationType	For process server-specific components, set to `ReportSituation`.
reasoningScope	For process server-specific components, set to `EXTERNAL`.

Business objects in events

Service components under certain circumstances process business objects, and monitoring can be configured to capture the business object data in Common Base Events. The business object data is encoded in XML elements, but these are converted to binary format before it is passed to the event. The data is encapsulated in the event in an extended data element, and stored in hexadecimal format.

You specify the level of business object detail that will be captured in Common Base Events. This level of detail affects only the amount of business object code that will be passed to the event; all of the other Common Base Event elements (both standard and event-specific) will be published to the event. The names of the detail levels applicable to Common Base Events differ depending on whether you created a static monitor using WebSphere Integration Developer, or a dynamic monitor on the administrative console, but they correspond as shown in the table below:

Administrative console detail level	Common Base Event/WebSphere Integration Developer detail level	Payload information published
FINE	EMPTY	None.
FINER	DIGEST	Payload description only.
FINEST	FULL	All of the payload.

The detail level is included in the event in an extended data element named `PayloadType`, and uses the Common Base Event names detailed above. The business object data itself is also included in the Common Base Event under an extended data element group with the name of the event extended data element appended by ″_BO.″ For example, if the business rule component fires an event with a nature of EXIT (WBI.BR.EXIT), the business object code is passed to an extended data element named `result`. Consequently, an extended data element named `result_BO` will be created in the event, which will then spawn child elements that will contain the actual business object data. The extended data element tree that is created for the business object code depends on the level of payload detail you specified for the monitor, as shown in this table:

Extended data element name	Type
FULL/FINEST	
<element_name>_BO	N/A
<element_name>_BO/TNS	String
<element_name>_BO/TYPE	String
<element_name>_BO/Raw Data	HexBinary
DIGEST/FINER	
<element_name>_BO	N/A
<element_name>_BO/TNS	String
<element_name>_BO/TYPE	String
<element_name>_BO/Verb (business object wrapped by a business graph only)	String

Extended data element name	Type
<element_name>_BO/Properties (business object wrapped by a business graph only)	HexBinary
EMPTY/FINE	
<element_name>_BO	N/A
<element_name>_BO/TNS	String
<element_name>_BO/TYPE	String
<element_name>_BO/Verb (business object wrapped by a business graph only)	String

The actual business object data is included in the event only if the monitor is set to record `FULL/FINEST` detail. The data is first serialized to XML format, but is then passed to the event named `Raw Data` in hexBinary format. An encoder/decoder is included with the process server to convert the serialized XML business object data to hexBinary, and back to XML, but the hexBinary format is what is actually stored in the event. If you are publishing the event output to the logger, then you will see the hexBinary output when you view the log files. If the event is published to the CEI server, then you can see the original XML format by using the Common Base Event browser to view the event.

This is a business object captured from a service component monitor, after the data was serialized to XML format by WebSphere Process Server:

```
<?xml version="1.0" encoding="UTF-8"?>
<mon:MonitorWrapper xsi:type="claim:Claim1BG"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:claim="http://Claim_Module"
  xmlns:mon="http://www.ibm.com/xmlns/prod/websphere/monitoring/6.0.0/mon">
  <Claim1>
    <employeeCode>ibm</employeeCode>
    <policyNumber>123</policyNumber>
    <situation>1</situation>
  </Claim1>
</mon:MonitorWrapper>
```

The data is first converted to hexBinary, and then passed to the event in an extended data element named `Raw Data`. The entire extended data element tree containing the encapsulated business object, including the `TNS` and `TYPE` elements, will be passed to the event as follows:

```
<extendedDataElements name="Claim1BG_BO" type="no value">
     <children name="TNS" type="string">
       <values>http://Claim_Module</values>
     </children>
     <children name="TYPE" type="string">
       <values>Claim1BG</values>
     </children>
     <children name="Raw Data" type="hexBinary">
       <hexValue>3C3F786D6C2076657273696F6E3D22312E302220656E636F64696E6
       73D225554462D38223F3E0D0A3C6D6F6E3A4D6F6E69746F725772617070657220
       7873693A747970653D22636C61696D3A436C61696D3142472220786D6C6E733A7
       873693D22687474703A2F2F7777772E77332E6F72672F323030312F584D4C5363
       68656D612D696E7374616E63652220786D6C6E733A636C61696D3D22687474703
       A2F2F436C61696D5F4D6F64756C652220786D6C6E733A6D6F6E3D22687474703A
       2F2F7777772E69626D2E636F6D2F786D6C6E732F70726F642F77656273706865
       2652F6D6F6E69746F72696E672F362E302E302F6D6F6E223E0D0A20203C436C61
       696D313E0D0A202020203C656D706C6F796565436F64653EE0B982E0B89BE0B8A
       3E0B981E0B881E0B8A3E0B8A1E0B897E0B8B5E0B988E0B88AE0B988E0B8A7E0B8
       A23C2F656D706C6F796565436F64653E0D0A202020203C706F6C6963794E756D6
```

```
        265723E3132333C2F706F6C6963794E756D6265723E0D0A202020203C73697475
        6174696F6E3E313C2F73697475617469696F6E3E0D0A20203C2F436C61696D313E0
        D0A3C2F6D6F6E3A4D6F6E69746F72577261707065723E0D0A
        </hexValue>
      </children>
</extendedDataElements>
```

Business Process Choreographer events

WebSphere Process Server incorporates the Business Process Choreographer service components for business processes and human tasks. The event points that can be monitored in these components are described in this section.

Situation-independent event data

Situation-independent event data for Common Base Event elements and for source component identifiers is shown in the following table.

Attribute	Description
	CommonBaseEvent element
creationTime	The time at which the event is created.
globalInstanceId	The identifier of the Common Base Event instance. This ID is automatically generated.
sequenceNumber	This value is set by the event factory.
severity	The impact that the event has on business processes or on human tasks. This attribute is set to `10` (information).
version	Set to `1.0.1`.
extensionName	The value depends on the object that creates the event and on the event.
	sourceComponentId element
component	**For business processes** Set to the identification of the current platform, followed by the version identification of the underlying software stack. **For human tasks** Set to `WPS`, followed by the version number of the software stack.
componentIdType	Set to `Component Kind QName`.
executionEnvironment	A string that identifies the operating system.
instanceId	The identifier of the server. This identifier has the format *`cell name`*`/`*`node name`*`/`*`server name`*. The delimiters are platform dependent.
location	Set to the host name of the executing server.
locationType	Set to `Hostname`.
processId	The process identifier of the operating system.
subcomponent	For business processes, set to `BFM`. For human tasks, set to `HTM`.
threadId	The thread identifier of the Java Virtual Machine (JVM).

Attribute	Description
componentType	For business processes, set to: www.ibm.com/namespaces/autonomic/Workflow_Engine For human tasks, set to: www.ibm.com/xmlns/prod/websphere/scdl/human-task

Monitoring business processes

Attributes relating to business processes are described here.

This topic describes the attributes that are associated with business processes: their event data, situations, and events.

Event data specific to business processes:

In business processes, processes, activities, scopes, links, and variables can send event data. The object-specific content of each of these event types is described here.

If not specified otherwise, the object-specific content is written as *extendedDataElement* XML elements of type string.

Process

Events of process instances have the following object-specific event content:

Attribute	Description
processTemplateName	The name of the process template from which the instance was derived
processTemplateValidFrom	The date from which the template is valid
processTemplateId	The identifier of the process template
processInstanceDescription	Optional: The description of the process instance
processInstanceExecutionState	A string representation of the state of the process
PayloadType	The string full

Activity and scope

Activities and scopes have the following object-specific event content:

Attribute	Description
processTemplateName	The name of the process template from which the instance was derived
processTemplateValidFrom	The date from which the template is valid
activityTemplateName	Optional: The name of the activity template from which the instance was derived
activityInstanceDescription	Optional: The description of the activity instance
activityKind	A string value that identifies the activity kind. This value has the format: *kind number-kind description*

Attribute	Description
state	A string value that represents the state of the activity. It has the format: *state number-state description*
bpelId	A string value that represents the wpc:id attribute of the activity
PayloadType	The string full

Link

Links have the following object-specific event content:

Attribute	Description
processTemplateName	The name of the process template from which the instance was derived
processTemplateValidFrom	The date from which the template is valid
flowBpelId	A string value that represents the wpc:id attribute of the flow activity that contains the link
elementName	The name of the link that was evaluated
description	Optional: A description of the link
PayloadType	The string full

Variable

Variables have the following object-specific event content.

Attribute	Description
processTemplateName	The name of the process template from which the instance was derived.
processTemplateValidFrom	The date from which the template is valid.
variableName	The name of the variable that was changed.
variableData	An XML representation of the content of the variable. The XML fragment is written into an extended data element of type hexBinary.
bpelId	A string value that represents the wpc:id attribute of the activity.
PayloadType	The string full.

Situations in business process events:

The CommonBaseEvent element contains elements that give more information about the situation that caused the event.

Business process events have the following situations:

Situation name	**Content of the Common Base Event**	
Start	categoryName is set to StartSituation.	
	situationType	
	Type	StartSituation
	reasoningScope	EXTERNAL
	successDisposition	SUCCESSFUL
	situationQualifier	START_COMPLETED
Stop	categoryName is set to StopSituation.	
	situationType	
	Type	StopSituation
	reasoningScope	EXTERNAL
	successDisposition	SUCCESSFUL
	situationQualifier	STOP_COMPLETED
Destroy	categoryName is set to DestroySituation.	
	situationType	
	Type	DestroySituation
	reasoningScope	EXTERNAL
	successDisposition	SUCCESSFUL
Fail	categoryName is set to StopSituation.	
	situationType	
	Type	StopSituation
	reasoningScope	EXTERNAL
	successDisposition	UNSUCCESSFUL
	situationQualifier	STOP_COMPLETED
Report	categoryName is set to ReportSituation.	
	situationType	
	Type	ReportSituation
	reasoningScope	EXTERNAL
	reportCategory	STATUS

Business process events:

Business process events are sent if monitoring is requested for the business process elements in WebSphere Integration Developer.

The following table shows the business process events and their situation-dependent data.

The columns are as follows:

Code Contains the number of the event. The value is written to the Common Base Event as an extended data element with the name BPCEventCode.

Extension name
Contains the string value that is used as the value of the extensionName attribute of the Common Base Event.

Situation
Refers to the situation name of the business process event. For details of situations, see "Situations in business process events" on page 22.

Event nature
Points, in the EventNature parameter, to the selectable event ″situations″ of a business process element, as they appear in WebSphere Integration Developer.

ECS current ID
ECS parent ID
Contain, in the ECSCurrentID and ECSParentID parameters, the identifiers of the event correlation spheres that are used. These values are written to the Common Base Event as context data elements.

Code	Description	Extension name	Situation	Event nature	ECS current ID	ECS parent ID
Process events						
21000	Process started	BPC.BFM.PROCESS.START[5]	Start	ENTRY	ID of the process instance	Not specified[1]
21001	Process suspended	BPC.BFM.PROCESS.STATUS	Report	SUSPENDED	ID of the process instance	Not specified[1]
21002	Process resumed	BPC.BFM.PROCESS.STATUS	Report	RESUMED	ID of the process instance	Not specified[1]
21004	Process completed	BPC.BFM.PROCESS.STATUS	Stop	EXIT	ID of the process instance	Not specified[1]
21005	Process terminated	BPC.BFM.PROCESS.STATUS	Stop	TERMINATED	ID of the process instance	Not specified[1]
21019	Process restarted	BPC.BFM.PROCESS.STATUS	Stop	RESTARTED	ID of the process instance	Not specified[1]
21020	Process deleted	BPC.BFM.PROCESS.STATUS	Destroy	DELETED	ID of the process instance	Not specified[1]
42001	Process failed[4]	BPC.BFM.PROCESS.FAILURE[4]	Fail	FAILED	ID of the process instance	Not specified[1]
42003	Process compensating	BPC.BFM.PROCESS.STATUS	Report	COMPENSATING	ID of the process instance	Not specified[1]
42004	Process compensated	BPC.BFM.PROCESS.STATUS	Stop	COMPENSATED	ID of the process instance	Not specified[1]
42009	Process terminating	BPC.BFM.PROCESS.STATUS	Report	TERMINATING	ID of the process instance	Not specified[1]

Code	Description	Extension name	Situation	Event nature	ECS current ID	ECS parent ID
42010	Process failing	BPC.BFM.PROCESS.STATUS	Report	FAILING	ID of the process instance	Not specified[1]
42027	Correlation set initialized	BPC.BFM.PROCESS.CORREL[7]	Report	CORRELATION	ID of the process instance	Not specified[1]
42041	Process work item deleted	BPC.BFM.PROCESS. WISTATUS[5]	Report	WI_DELETED	ID of the process instance	Not specified[1]
42042	Process work item created	BPC.BFM.PROCESS. WISTATUS[5]	Report	WI_CREATED	ID of the process instance	Not specified[1]
42046	Process compensation failed	BPC.BFM.PROCESS.STATUS	Fail	COMPFAILED	ID of the process instance	Not specified[1]
42047	Process event received	BPC.BFM.PROCESS.STATUS	Report	EV_RECEIVED	ID of the process instance	Not specified[1]
42049	Process event escalated	BPC.BFM.PROCESS.STATUS	Report	EV_ESCALATED	ID of the process instance	Not specified[1]
			Activity events			
21006	Activity ready	BPC.BFM.ACTIVITY.STATUS	Start	CREATED	ID of the activity	ID of the containing scope or process
21007	Activity started for an invoke operation	BPC.BFM.ACTIVITY.STATUS BPC.BFM.ACTIVITY.MESSAGE[6]	Start	ENTRY	ID of the activity	ID of the containing scope or process
21011	Activity completed for an invoke, staff, receive, or reply operation	BPC.BFM.ACTIVITY.STATUS BPC.BFM.ACTIVITY.MESSAGE[6]	Stop	EXIT	ID of the activity	ID of the containing scope or process
21021	Claim canceled	BPC.BFM.ACTIVITY.STATUS	Report	DEASSIGNED	ID of the activity	ID of the containing scope or process
21022	Activity claimed	BPC.BFM.ACTIVITY.CLAIM[2]	Report	ASSIGNED	ID of the activity	ID of the containing scope or process
21027	Activity terminated	BPC.BFM.ACTIVITY.STATUS	Stop	TERMINATED	ID of the activity	ID of the containing process instance

Code	Description	Extension name	Situation	Event nature	ECS current ID	ECS parent ID
21080	Activity failed	BPC.BFM.ACTIVITY.FAILURE[3]	Failed	FAILED	ID of the activity	ID of the containing scope or process
21081	Activity expired	BPC.BFM.ACTIVITY.STATUS	Report	EXPIRED	ID of the activity	ID of the containing scope or process
42005	Activity skipped	BPC.BFM.ACTIVITY.STATUS	Report	SKIPPED	ID of the activity	ID of the containing scope or process
42012	Activity output message set	BPC.BFM.ACTIVITY.MESSAGE[6]	Report	OUTPUTSET	ID of the activity	ID of the containing scope or process
42013	Activity fault message set	BPC.BFM.ACTIVITY.MESSAGE[6]	Report	FAULTSET	ID of the activity	ID of the containing scope or process
42015	Activity stopped	BPC.BFM.ACTIVITY.STATUS	Stop	STOPPED	ID of the activity	ID of the containing scope or process
42031	Activity force retried	BPC.BFM.ACTIVITY.STATUS	Report	FRETRIED	ID of the activity	ID of the containing scope or process
42032	Activity force completed	BPC.BFM.ACTIVITY.STATUS	Stop	FCOMPLETED	ID of the activity	ID of the containing scope or process
42036	Activity has message received	BPC.BFM.ACTIVITY.MESSAGE[6]	Report	EXIT	ID of the activity	ID of the containing scope or process
42037	Loop condition true	BPC.BFM.ACTIVITY.STATUS	Report	CONDTRUE	ID of the activity	ID of the containing scope or process
42038	Loop condition false	BPC.BFM.ACTIVITY.STATUS	Report	CONDFALSE	ID of the activity	ID of the containing scope or process
42039	Work item deleted	BPC.BFM.ACTIVITY. WISTATUS[5]	Report	WI_DELETED	ID of the activity	ID of the containing scope or processID of the containing scope or process

Code	Description	Extension name	Situation	Event nature	ECS current ID	ECS parent ID
42040	Work items created	BPC.BFM.ACTIVITY. WISTATUS[5]	Report	WI_CREATED	ID of the activity	ID of the containing scope or process
42050	Activity escalated	BPC.BFM.ACTIVITY.STATUS	Report	ESCALATED	ID of the activity	ID of the containing scope or process
42054	Activity work items refreshed	BPC.BFM.ACTIVITY. WISTATUS[5]	Report	WI_REFRESHED	ID of the activity	ID of the containing scope or process
42055	Work item transferred	BPC.BFM.ACTIVITY. WITRANSFER[8]	Report	WI_TRANSFERRED	ID of the activity	ID of the containing scope or process
Scope events						
42020	Scope started	BPC.BFM.ACTIVITY.STATUS	Start	ENTRY	ID of the scope	ID of the containing scope or process
42021	Scope skipped	BPC.BFM.ACTIVITY.STATUS	Report	SKIPPED	ID of the scope	ID of the containing scope or process
42022	Scope failed	BPC.BFM.ACTIVITY.FAILURE[3]	Fail	FAILED	ID of the scope	ID of the containing scope or process
42023	Scope terminating	BPC.BFM.ACTIVITY.STATUS	Report	TERMINATING	ID of the scope	ID of the containing scope or process
42024	Scope terminated	BPC.BFM.ACTIVITY.STATUS	Stop	TERMINATED	ID of the scope	ID of the containing scope or process
42026	Scope completed	BPC.BFM.ACTIVITY.STATUS	Stop	EXIT	ID of the scope	ID of the containing scope or process
42043	Scope compensating	BPC.BFM.ACTIVITY.STATUS	Report	COMPENSATING	ID of the scope	ID of the containing scope or process
42044	Scope compensated	BPC.BFM.ACTIVITY.STATUS	Stop	COMPENSATED	ID of the scope	ID of the containing scope or process

Code	Description	Extension name	Situation	Event nature	ECS current ID	ECS parent ID
42044	Scope compensation failed	BPC.BFM.ACTIVITY.STATUS	Fail	COMPFAILED	ID of the scope	ID of the containing scope or process
42048	Scope event received	BPC.BFM.ACTIVITY.STATUS	Report	EV_RECEIVED	ID of the scope	ID of the containing scope or process
42051	Scope event escalated	BPC.BFM.ACTIVITY.STATUS	Report	EV_ESCALATED	ID of the scope	ID of the containing scope or process
Link events						
21034	Link evaluated true	BPC.BFM.LINK.STATUS	Report	CONDTRUE	ID of the containing flow activity	ID of the containing scope or process
42000	Link evaluated false	BPC.BFM.LINK.STATUS	Report	CONDFALSE	ID of the containing flow activity	ID of the containing scope or process
Variable event						
21090	Variable update	BPC.BFM.VARIABLE.STATUS	Report	CHANGED	ID of the containing scope or process	Not specified[1]

Code	Description	Extension name	Situation	Event nature	ECS current ID	ECS parent ID

1. The ECSParentID is the ECSCurrentID before the process instance start event of the current process.
2. The following extended data elements are added to the event:

   username
   The user who claimed the activity

   principal
   The principal who initiated the claim
3. The following extended data element is added to the event:

   activityFailedException
   The exception that caused the activity to fail
4. The following extended data element is added to the event:

   processFailedException
   The exception that caused the process to fail
5. The following extended data element is added to the event:

   userName
   The names of the users who are associated with the event
6. The following extended data element is added to the event:

   message The input or the output message as an XML fragment of type hexBinary
7. The following extended data element is added to the event:

   correlationSet
   The content is added as an XML fragment of type hexBinary
8. The following extended data elements are added to the event:

   current The name of the current owner of the work item

   target The name of the new owner of the work item

Monitoring human tasks

Attributes relating to human tasks are described here.

This topic describes the attributes that are associated with human tasks: their event data, situations, and events.

Event data specific to human tasks:

Events are created on behalf of tasks and escalations. The object-specific content of each of these event types is described here.

Tasks

If not specified otherwise, the content is written as extendedDataElements of type string.

Task events have the following object-specific event content.

Attribute	Description
taskTemplateName	The name of the task template from which the instance was derived
taskTemplateValidFrom	The date from which the template is valid

Attribute	Description
taskTemplateId	The identifier of the task template from which the instance is derived
taskInstanceDescription	If this attribute is set, it is the description of the task instance
PayloadType	The string full

Escalation

Escalations have the following object-specific event content:

Attribute	Description
taskTemplateName	The name of the task template from which the instance was derived
taskTemplateValidFrom	The date from which the template is valid
taskTemplateId	The identifier of the task template from which the instance is derived
taskInstanceDescription	Optional: The description of the activity instance
escalationName	The name of the escalation
escalationInstanceDescription	Optional: The description of the escalation instance
PayloadType	The string full

Situations in human task events:

The CommonBaseEvent element contains elements that give more information about the situation that caused the event.

Human task events have the following situations:

Situation name	Content of the Common Base Event	
Start	categoryName is set to `StartSituation`.	
	situationType	
	Type	`StartSituation`
	reasoningScope	`EXTERNAL`
	successDisposition	`SUCCESSFUL`
	situationQualifier	`START_COMPLETED`
Stop	categoryName is set to `StopSituation`.	
	situationType	
	Type	`StopSituation`
	reasoningScope	`EXTERNAL`
	successDisposition	`SUCCESSFUL`
	situationQualifier	`STOP_COMPLETED`

Situation name	Content of the Common Base Event	
Destroy	categoryName is set to `DestroySituation`.	
	situationType	
	Type	`DestroySituation`
	reasoningScope	`EXTERNAL`
	successDisposition	`SUCCESSFUL`
Fail	categoryName is set to `StopSituation`.	
	situationType	
	Type	`StopSituation`
	reasoningScope	`EXTERNAL`
	successDisposition	`UNSUCCESSFUL`
	situationQualifier	`STOP_COMPLETED`
Report	categoryName is set to `ReportSituation`.	
	situationType	
	Type	`ReportSituation`
	reasoningScope	`EXTERNAL`
	reportCategory	`STATUS`

Human task events:

Human task events are sent if monitoring is requested for the elements of the task in WebSphere Integration Developer.

The following table shows the human task events and their situation-dependent data.

The **Code** column contains the number of the event. The value is written to the Common Base Event as an extended data element with the name HTMEventCode. The columns are as follows:

Extension name
: Contains the string value that is used as the value of the extensionName attribute of the Common Base Event.

Situation
: Refers to the situation name of the business process event. For details of situations, see “Situations in human task events” on page 30.

Event nature
: Points, in the EventNature parameter, to the selectable event ″situations″ of a human task element, as they appear in WebSphere Integration Developer.

ECS current ID
ECS parent ID
: Contain, in the ECSCurrentID and ECSParentID parameters, the identifiers of the event correlation spheres that are used. These values are written to the Common Base Event as context data elements.

Code	Description	Extension name	Situation	Event nature	ECS current ID	ECS parent ID
Task events						
51001	Task created	BPC.HTM.TASK. INTERACT[2]	Report	CREATED	ID of the task instance	Not specified[1]
51002	Task deleted	BPC.HTM.TASK.STATUS	Destroy	DELETED	ID of the task instance	Not specified[1]
51003	Task started	BPC.HTM.TASK.STATUS	Start	ENTRY	ID of the task instance	Not specified[1]
51004	Task completed	BPC.HTM.TASK.STATUS	Stop	EXIT	ID of the task instance	Not specified[1]
51005	Claim canceled	BPC.HTM.TASK.STATUS	Report	DEASSIGNED	ID of the task instance	Not specified[1]
51006	Task claimed	BPC.HTM.TASK. INTERACT[2]	Report	ASSIGNED	ID of the task instance	Not specified[1]
51007	Task terminated	BPC.HTM.TASK.STATUS	Stop	TERMINATED	ID of the task instance	Not specified[1]
51008	Task failed	BPC.HTM.TASK. FAILURE[3]	Fail	FAILED	ID of the task instance	Not specified[1]
51009	Task expired	BPC.HTM.TASK.STATUS	Report	EXPIRED	ID of the task instance	Not specified[1]
51010	Waiting for subtasks	BPC.HTM.TASK.STATUS	Report	WAITFORSUBTASK	ID of the task instance	Not specified[1]
51011	Subtasks completed	BPC.HTM.TASK.STATUS	Stop	SUBTASKCOMPLETED	ID of the task instance	Not specified[1]
51012	Task restarted	BPC.HTM.TASK.STATUS	Report	RESTARTED	ID of the task instance	Not specified[1]
51013	Task suspended	BPC.HTM.TASK.STATUS	Report	SUSPENDED	ID of the task instance	Not specified[1]
51014	Task resumed	BPC.HTM.TASK.STATUS	Report	RESUMED	ID of the task instance	Not specified[1]
51015	Task completed and follow-on task started	BPC.HTM.TASK. FOLLOW[5]	Report	COMPLETEDFOLLOW	ID of the task instance	Not specified[1]
51101	Task properties updated	BPC.HTM.TASK.STATUS	Report	UPDATED	ID of the task instance	Not specified[1]
51103	Output message updated	BPC.HTM.TASK. MESSAGE[4]	Report	OUTPUTSET	ID of the task instance	Not specified[1]
51104	Fault message updated	BPC.HTM.TASK. MESSAGE[4]	Report	FAULTSET	ID of the task instance	Not specified[1]
51201	Work item deleted	BPC.HTM.TASK. WISTATUS[2]	Destroy	WI_DELETED	ID of the task instance	Not specified[1]
51202	Work items created	BPC.HTM.TASK. WISTATUS[2]	Report	WI_CREATED	ID of the task instance	Not specified[1]
51204	Work item transferred	BPC.HTM.TASK. WITRANSFER[6]	Report	WI_TRANSFERRED	ID of the task instance	Not specified[1]

Code	Description	Extension name	Situation	Event nature	ECS current ID	ECS parent ID
51205	Work items refreshed	BPC.HTM.TASK. WISTATUS[2]	Report	WI_REFRESHED	ID of the task instance	Not specified[1]
			Escalation events			
53001	Escalation fired	BPC.HTM.ESCALATION. STATUS	Report	ENTRY	ID of the escalation	ID of the associated task instance
53202	Work item created	BPC.HTM.ESCALATION. WISTATUS[2]	Report	WI_CREATED	ID of the escalation	ID of the associated task instance
53201	Work item deleted	BPC.HTM.ESCALATION. WISTATUS[2]	Destroy	WI_DELETED	ID of the escalation	ID of the associated task instance
53204	Escalation transferred	BPC.HTM.ESCALATION. WITRANSFER[6]	Report	WI_TRANSFERRED	ID of the escalation	ID of the associated task instance
53205	Work item refreshed	BPC.HTM.ESCALATION. WISTATUS[2]	Report	WI_REFRESHED	ID of the escalation	ID of the associated task instance

1. The ECSParentID is the ECSCurrentID before the task instance event.
2. The following extended data elements are added to the event:

   username
   The names of the users who are associated with the event
3. The following extended data element is added to the event:

   taskFailedException
   The exception that caused the task to fail
4. The following extended data element is added to the event:

   message The input or the output message as an XML fragment of type hexBinary
5. The following extended data element is added to the event:

   followTaskId
   The ID of the task that was started as a follow-on-task
6. The following extended data elements are added to the event:

   current The name of the current owner of the work item

   target The name of the new owner of the work item

Process server events

WebSphere Process Server features its own service components, and each of these components has its own set of event points that can be monitored. Service components contain one or more elements, which are sets of different steps processed in each service component. In turn, each element has its own set of event natures, that are key points that are reached when processing a service component element. This section describes all of the service components, their elements and associated event natures, and the extended data elements unique to each event.

Adapter events

The elements of the adapter component (base name `WBI.JCAAdapter`) that can be monitored are listed here, along with their associated event natures, event names, and the extended data elements that are unique to each event.

Event Name	Event Natures	Extended Data Elements	Type
`Polling` element			
WBI.JCAAdapter. Polling.Started	Started	PollFrequency	Int
		PollQuantity	Int
WBI.JCAAdapter. Polling.Stopped	Stopped	N/A	
`Delivery` element			
WBI.JCAAdapter. Delivery.EXIT	EXIT	Input_BO	See “Business objects in events” on page 18
		EventID	String
WBI.JCAAdapter. Delivery.FAILURE	FAILURE	Input_BO	See “Business objects in events” on page 18
		EventID	String
		FailureReason	exception
`Connection` element			
WBI.JCAAdapter. Connection.FAILURE	FAILURE	N/A	

Business rule events

The business rule component (base name `WBI.BR`) contains a single element that can be monitored. All event types for this element are listed here, with their associated event natures, event names, and the extended data elements that are unique to each event.

Event Name	Event Nature	Extended Data Elements	Type
WBI.BR.ENTRY	ENTRY	operationName	String
		input	See “Business objects in events” on page 18
WBI.BR.EXIT	EXIT	operationName	String
		result	See “Business objects in events” on page 18
WBI.BR.FAILURE	FAILURE	FailureReason	HexBinary (StackTrace)
		operationName	String

Event Name	Event Nature	Extended Data Elements	Type
WBI.BR. SelectionKeyExtracted	SelectionKeyExtracted	operationName	String
		input	See “Business objects in events” on page 18
WBI.BR.TargetFound	TargetFound	operationName	String
		target	String

Business state machine events

The elements from the business state machine component (base name `WBI.BSM`) that can be monitored are listed here, along with their associated event natures, event names, and all extended data elements that are unique to each event.

Event Name	Event Nature	Extended Data Elements	Type
`StateMachineDefinition` element			
WBI.BSM. StateMachineDefinition. ALLOCATED	ALLOCATED	instanceID	String
WBI.BSM. StateMachineDefinition. RELEASED	RELEASED	instanceID	String
`Transition` element			
WBI.BSM.Transition.ENTRY	ENTRY	instanceID	String
		name	String
		input	See “Business objects in events” on page 18
WBI.BSM.Transition.EXIT	EXIT	instanceID	String
		name	String
		output	See “Business objects in events” on page 18
WBI.BSM.Transition.FAILURE	FAILURE	FailureReason	HexBinary (StackTrace)
		instanceID	String
		name	String
		input	See “Business objects in events” on page 18
`State` element			
WBI.BSM.State.ENTRY	ENTRY	instanceID	String
		name	String
WBI.BSM.State.EXIT	EXIT	instanceID	String
		name	String
WBI.BSM.State.FAILURE	FAILURE	FailureReason	HexBinary (StackTrace)
		instanceID	String
		name	String

Event Name	Event Nature	Extended Data Elements	Type
Guard element			
WBI.BSM.Guard.ENTRY	ENTRY	instanceID	String
		name	String
WBI.BSM.Guard.EXIT	EXIT	instanceID	String
		name	String
		result	Boolean
WBI.BSM.Guard.FAILURE	FAILURE	FailureReason	HexBinary (StackTrace)
		instanceID	String
		name	String
Action element			
WBI.BSM.Action.ENTRY	ENTRY	instanceID	String
		name	String
WBI.BSM.Action.EXIT	EXIT	instanceID	String
		name	String
WBI.BSM.Action.FAILURE	FAILURE	FailureReason	HexBinary (StackTrace)
		instanceID	String
		name	String
EntryAction element			
WBI.BSM.EntryAction.ENTRY	ENTRY	instanceID	String
		name	String
WBI.BSM.EntryAction.EXIT	EXIT	instanceID	String
		name	String
WBI.BSM.EntryAction.FAILURE	FAILURE	FailureReason	HexBinary (StackTrace)
		instanceID	String
		name	String
ExitAction element			
WBI.BSM.ExitAction.ENTRY	ENTRY	instanceID	String
		name	String
WBI.BSM.ExitAction.EXIT	EXIT	instanceID	String
		name	String
WBI.BSM.ExitAction.FAILURE	FAILURE	FailureReason	HexBinary (StackTrace)
		instanceID	String
		name	String
Timer element			
WBI.BSM.Timer.START	START	instanceID	String
		name	String
		duration	String

Event Name	Event Nature	Extended Data Elements	Type
WBI.BSM.Timer.STOPPED	STOPPED	instanceID	String
		name	String
		duration	String

Map events

The elements from the map component (base name `WBI.MAP`) that can be monitored are listed here, along with their event natures, event names, and all extended data elements that are unique to each event.

Event Name	Event Nature	Extended Data Elements	Type
WBI.MAP.ENTRY	ENTRY	input	See "Business objects in events" on page 18
WBI.MAP.EXIT	EXIT	output	See "Business objects in events" on page 18
WBI.MAP.FAILURE	FAILURE	input	See "Business objects in events" on page 18
		output	See "Business objects in events" on page 18
		FailureReason	HexBinary (StackTrace)
`Transformation` element			
WBI.MAP.Transformation. ENTRY	ENTRY	input	See "Business objects in events" on page 18
WBI.MAP.Transformation. EXIT	EXIT	output	See "Business objects in events" on page 18
WBI.MAP.Transformation. FAILURE	FAILURE	input	See "Business objects in events" on page 18
		output	See "Business objects in events" on page 18
		FailureReason	HexBinary (StackTrace)

Mediation events

The elements from the mediation component (base name `WBI.MEDIATION`) that can be monitored are listed here, along with their associated event natures, names, and all extended data elements that are unique to each event.

Event Name	Event Nature	Extended Data Elements	Type
`OperationBinding` element			
WBI.MEDIATION. OperationBinding.ENTRY	ENTRY	Source	String
		Target	String
		input	See "Business objects in events" on page 18

Event Name	Event Nature	Extended Data Elements	Type
WBI.MEDIATION. OperationBinding.EXIT	EXIT	Source	String
		Target	String
		output	See "Business objects in events" on page 18
WBI.MEDIATION. OperationBinding.FAILURE	FAILURE	Source	String
		Target	String
		input (optional)	See "Business objects in events" on page 18
		FailureReason	HexBinary (StackTrace)
`ParameterMediation` element			
WBI.MEDIATION. ParameterMediation.ENTRY	ENTRY	Type	String
		TransformName	String
		input	See "Business objects in events" on page 18
WBI.MEDIATION. ParameterMediation.EXIT	EXIT	Type	String
		TransformName	String
		output	See "Business objects in events" on page 18
WBI.MEDIATION. ParameterMediation.FAILURE	FAILURE	Type	String
		TransformName	String
		input (optional)	See "Business objects in events" on page 18
		FailureReason	HexBinary (StackTrace)

Recovery events

The recovery component (base name `WBI.Recovery`) contains a single element that can be monitored. All event types for this element are listed here, along with their associated event natures, event names, and the extended data elements that are unique to each event.

Event Name	Event Nature	Extended Data Elements	Type
WBI.Recovery.FAILURE	FAILURE	MsgId	String
		DestModuleName	String
		DestComponentName	String
		DestMethodName	String
		SourceModuleName	String
		SourceComponentName	String
		ResubmitDestination	String
		ExceptionDetails	String
		SessionId	String
		FailureTime	dateTime
		ExpirationTime	dateTime
		Status	int
		MessageBody	byteArray
		Deliverable	boolean
WBI.Recovery.DEADLOOP	DEADLOOP	DeadloopMsgId	String
		SIBusName	String
		QueueName	String
		Reason	String
WBI.Recovery.RESUBMIT	RESUBMIT	MsgId	String
		OriginalMesId	String
		ResubmitCount	int
		Description	String
WBI.Recovery.DELETE	DELETE	MsgId	String
		deleteTime	dateTime
		Description	String

Service Component Architecture events

The Service Component Architecture contains a single element, with a base name of `WBI.JService.MethodInvocation`. All the events and associated natures of this element are listed here, along with all extended data elements and that are unique to each event.

Event Name	Event Nature	Extended Data Elements	Type
WBI.JService. MethodInvocation.ENTRY	ENTRY	SOURCE COMPONENT	String
		SOURCE INTERFACE	String
		SOURCE METHOD	String
		SOURCE MODULE	String
		SOURCE REFERENCE	String
		TARGET COMPONENT	String
		TARGET INTERFACE	String
		TARGET METHOD	String
		TARGET MODULE	String
		Input	See "Business objects in events" on page 18
WBI.JService. MethodInvocation.EXIT	EXIT	SOURCE COMPONENT	String
		SOURCE INTERFACE	String
		SOURCE METHOD	String
		SOURCE MODULE	String
		SOURCE REFERENCE	String
		TARGET COMPONENT	String
		TARGET INTERFACE	String
		TARGET METHOD	String
		TARGET MODULE	String
		Input	See "Business objects in events" on page 18

Event Name	Event Nature	Extended Data Elements	Type
WBI.JService. MethodInvocation.FAILURE	FAILURE	SOURCE COMPONENT	String
		SOURCE INTERFACE	String
		SOURCE METHOD	String
		SOURCE MODULE	String
		SOURCE REFERENCE	String
		TARGET COMPONENT	String
		TARGET INTERFACE	String
		TARGET METHOD	String
		TARGET MODULE	String
		FailureReason	HexBinary (StackTrace)
		Input	See "Business objects in events" on page 18

Selector events

The selector component contains a single element that can be monitored. All event types for this element are listed here, along with their associated event natures, event names, and the extended data elements that are unique to each event. All selector events have a base name of `WBI.SEL`.

Event Name	Event Nature	Extended Data Elements	Type
WBI.SEL.ENTRY	ENTRY	operationName	String
		input	See "Business objects in events" on page 18
WBI.SEL.EXIT	EXIT	operationName	String
		result	See "Business objects in events" on page 18
WBI.SEL.FAILURE	FAILURE	FailureReason	HexBinary (StackTrace)
		operationName	String
WBI.SEL. SelectionKeyExtracted	SelectionKeyExtracted	operationName	String
		input	See "Business objects in events" on page 18
WBI.SEL.TargetFound	TargetFound	operationName	String
		target	String

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:

IBM World Trade Asia Corporation Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, 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 Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

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

Programming interface information

Programming interface information, if provided, is intended to help you create application software using this program.

General-use programming interfaces allow you to write application software that obtain the services of this program's tools.

However, this information may also contain diagnosis, modification, and tuning information. Diagnosis, modification and tuning information is provided to help you debug your application software.

Warning: Do not use this diagnosis, modification, and tuning information as a programming interface because it is subject to change.

Trademarks and service marks

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

i5/OS
IBM
the IBM logo
AIX
AIX 5L
CICS
CrossWorlds
DB2
DB2 Universal Database
Domino
HelpNow
IMS
Informix
iSeries
Lotus
Lotus Notes
MQIntegrator
MQSeries
MVS
Notes
OS/390
OS/400
Passport Advantage
pSeries
Redbooks
SupportPac
WebSphere
z/OS

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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

WebSphere Process Server, Version 6.0

Printed in USA