CA Application Performance Management - 10.1

Monitor webMethods Integration Server

Date: 03-Nov-2016

Table of Contents

About webMethods Integration Server 7

How to Enable Monitoring for webMethods Integration Server 10

Manually Enable the Agent for Monitoring webMethods Integration Server 10

About the Directive Files for webMethods Integration Server 12

Enable the Enterprise Manager Extension 12

Use Dashboards to Monitor webMethods 14

Filtering the services monitored and displayed 17

About the Default Configuration File 17

Excluding services using regular expressions 17

Specifying a Different Location for the Configuration File 18

View and Navigate Metrics for webMethods 19

Metrics for Adapters 19

- Metrics for Adapter Connection Pools 20
- Metrics for Adapter Notifications 21
- Metrics for Adapter Services 21

Metrics for Authorization 21

Metrics for Business Processes 22

- View Step-Level Metrics for Business Processes 23
- About Average Response Time for completed processes 23
- About Responses Per Interval for completed processes 24
- About Errors Per Interval for failed processes 24
- About Concurrent Invocations for steps in a process 24

Metrics for Flow Services 24

- Flow Services That Call Other Flow Services 24

Metrics for Java Services 25

Metrics for JDBC Pools 25

Metrics for Thread Pools 26
Metrics for Trading Networks 26
Metrics for Document Types 27
Metrics for Processing Rules 27
About Average Response Time for Processing Rules 28
Metrics for Service Execution Tasks 28
Metrics for Service Threads 28
Metrics for Triggers 28
Metrics for WebServices 29
About server-side metrics 29
About client-side metrics 29
Metrics for XSLT Services 30

Viewing default webMethods metric groupings 31

Viewing default webMethods alerts 32

Viewing webMethods dependencies 33

Tracing transactions for webMethods 34
About Configuring Cross-Process Transaction Tracing 34
Understanding the value of cross-process transaction tracing 35
Starting and viewing a sample transaction trace 35
Using the Sequence View for webMethods processes 35

Monitor webMethods Integration Server

The Software AG webMethods product suite is a SOA platform that consists of multiple infrastructure components and features to enable organizations to create, orchestrate, and integrate business processes and web services. The SOA extension for webMethods enables you to monitor many key elements of the webMethods infrastructure.

This article describes webMethods services and the related dashboards and metrics you can use to monitor and analyze the health and operation of the webMethods Integration Server.

- About webMethods Integration Server (see page 7)
- How to Enable Monitoring for webMethods Integration Server (see page 10)
- Use Dashboards to Monitor webMethods (see page 14)
- Filtering the services monitored and displayed (see page 17)
- View and Navigate Metrics for webMethods (see page 19)
- Viewing default webMethods metric groupings (see page 31)
- Viewing default webMethods alerts (see page 32)
- Viewing webMethods dependencies (see page 33)
- Tracing transactions for webMethods (see page 34)

About webMethods Integration Server

The webMethods Integration Server enables companies to expose and integrate new and existing business services. It includes tools that enable you to design, test, and deploy new services, and automate, orchestrate, and assemble loosely-coupled services and legacy systems into improved business processes.

The webMethods Integration Server provides a centralized platform for running and distributing services. It receives and interprets client requests, identifies and invokes the requested services, passes data to running services in the expected format, receives the output produced by the services, and returns output to the clients.

As an integration platform, webMethods is used primarily for coordinating operation between application servers, databases, and custom applications and enabling companies or trading partners to exchange electronic documents.

You can monitor the operation of webMethods Integration Server with metrics for the following top-level components:

- **Adapters**
  Adapters enable external applications to integrate with webMethods through a common adapter framework of service interfaces. Adapters consist of:
  Adapter Connections that enable Integration Server to connect to the external resources or systems at run-time.
  Adapter Services that run on Integration Server initiate operations on the external resource.
  Adapter Notifications that monitor the external resource and notify the Integration Server when an event not initiated by the Integration Server has occurred.
  With the SOA extension for webMethods, you can monitor the performance and overall health for all adapters you have depolyed for webMethods Integration Server using the metrics under the Adapter Connection Pools, Adapter Services, and Adapter Notifications nodes.

- **Business Processes**
  A business process is a series of interrelated business tasks that are performed in a specific order, using a specific set of business rules. Most business processes also require the interaction of multiple systems and multiple people in different roles.
  For example, you may have business processes for preparing for a new employee, handling a purchase order, or, delivering invoices. Each business process then might include business tasks such as assigning office space for the new employee, adding the employee to the Human Resources (HR) system, and ordering office equipment and supplies.
  With the SOA extension for webMethods, you can monitor the performance and overall health for the business processes your have defined using the metrics under the WebMethods > Business Processes node.

- **Flow Services**
  Flow Services are services written in webMethods Flow language and deployed on the webMethods Integration Server. Flow services can invoke any service running on the webMethods server, including other flow services, user-defined services, built-in services, and services from other providers such as webMethods Adapters or .NET Plug-in.
  With the SOA extension for webMethods, you can monitor the performance and overall health for all flow services or filter the flow services to exclude flow services you are not interested in monitoring. Metrics for the individual flow steps in a flow service as listed under the WebMethods > Flow Services node.

- **Java Services**
  Java Services are built-in and user-defined services written in Java, or written in other languages and wrapped using a Java class, and exposed as services on the webMethods Integration Server.
  With the SOA extension for webMethods, you can monitor the performance and overall health for all Java services or filter the Java services to exclude the Java services you are not interested in monitoring. Metrics for the Java methods in each Java class are listed under the WebMethods > Java Services node.

- **JDBC Connection Pools**
  The webMethods Integration Server uses Java database connections to communicate and transfer information over the network.
  With the SOA extension for webMethods, you can monitor the availability of JDBC connections using the metrics under the WebMethods > JDBC Connection Pools node.

- **Thread Pools**
  The webMethods Integration Server uses threads to execute services, retrieve documents from the webMethods Broker, and execute triggers.
  With the SOA extension for webMethods, you can monitor the availability of threads using the metrics under the WebMethods > Thread Pools node.

- **Trading Networks**
  Trading networks enable organizations to exchange documents to establish and enrich business-to-business relationships.
  With the SOA extension for webMethods, you can monitor document recognition and processing using the metrics under the WebMethods > Trading Networks node.

- **Triggers**
  Triggers establish subscriptions to publishable document types and specify how to process instances of those documents.
  A Broker or Local trigger is a trigger that subscribes to and processes documents published locally on the Integration Server or delivered to the Broker. Broker Triggers are often associated with asynchronous Adapter Notifications.
  A JMS trigger is a trigger that receives messages from a destination (queue or topic) on a JMS provider and then processes those messages.
  With the SOA extension for webMethods, you can monitor triggers using the metrics under the WebMethods > Triggers node.

- **WebServices**
  WebServices metrics represent client and server business service endpoint and associated operations within each service.
  With the SOA extension for webMethods, you can monitor the performance and overall health of client and server web service endpoints under the WebMethods > WebServices node.

- **XSLT Services**
  Within webMethods, you can use XSLT stylesheets to transform XML data into different formats and include the transformation in other services.
  With the SOA extension for webMethods, you can monitor the performance and overall health of XSL transformations under the WebMethods > XSLT Services node.

How to Enable Monitoring for webMethods Integration Server

As the administrator, you enable monitoring for webMethods Integration Server by performing the following high-level steps:

1. Verify that you have webMethods Integration Server installed.
2. Verify whether you have a supported version of the webMethods WmPRT.jar file. Open the following URL in a web browser. Verify the version level listed.

   ```
   http://<Integration_Server_Hostname>:<port_number>/WmRoot/Updates.dsp
   ```

3. Verify that the agent and CA APM for SOA are installed and enabled.
4. Enable the agent to use CA APM for webMethods Integration Server by configuring the agent profile.

   Important! Skip this step if you enabled CA APM for webMethods Integration Server in the agent using the Standalone agent installer or using a response file.

5. Enable the Enterprise Manager extension (see page 12).

Manually Enable the Agent for Monitoring webMethods Integration Server

You can enable monitoring for webMethods Integration using one of the following methods:

- Select CA APMfor webMethods Integration Server when you install the agent.
  The agent profile is automatically configured with default settings. No further steps are required.
- Do not select CA APM for webMethods Integration Server when you install the agent. Configure the agent profile manually using the following procedure.

Follow these steps:

1. Verify that the Default agent is installed and enabled.

2. Copy the files from the <*Agent_Home*>/examples/SOAExtensionForWebMethodsIS directory to the corresponding <*Agent_Home*>/core directory.
   For example, copy the files in <*Agent_Home*>/examples/SOAExtensionForWebMethodsIS to the <*Agent_Home*>/core directory.

3. Open the <*Agent_Home*>/core/config/IntroscopeAgent.profile file in a text editor.

   a. Add the webmethods.pbl and the default agent pbl to the introscope.autoprobe.directivesFile property in the IntroscopeAgent.profile file. For example:

      ```
      introscope.autoprobe.directivesFile=default-typical.pbl,hotdeploy,
      webmethods.pbl
      ```

      You can customize tracing using the ProbeBuilder directive files for webMethods Integration Server (see page 12).

   b. Set the introscope.agent.agentName property value to webMethods Agent. For example:

      ```
      introscope.agent.agentName=webMethods Agent
      ```

      This step ensures that only webMethods data is displayed in dashboards.

   c. Modify any additional properties in the IntroscopeAgent.profile file if you want to change the default settings.

   d. Save and close the IntroscopeAgent.profile file.

4. Only valid for webMethods Integration Server 7.x, perform the following edits in the <*Agent_Home*>/core/config/webmethods-toggles.pbd file:

   a. Uncomment the BProcAndStepTracing71 flag in the Business Processes section.

      ```
      Turn on BProcAndStepTracing71 flag.
      Turn off BProcAndStepTracing80 flag.
      Turn off BProcAndStepTracing82 flag.
      ```

   b. Uncomment the following options in the Web Services section:

      ```
      Turn off WM8xWebServicesTracing flag.
      Turn on WM7xWebServicesTracing flag.
      Turn off WebMethods8xWSClientTracing flag.
      Turn on WebMethods7xWSClientTracing flag.
      Turn off WebMethods65WSClientTracing flag.
      ```

   c. Save and close the webmethods-toggles.pbd file.

5. Restart the webMethods Integration Server process.

About the Directive Files for webMethods Integration Server

When you set the introscope.autoprobe.directivesFile property in the IntroscopeAgent.profile, you enable the default instrumentation for webMethods Integration Server. You can then customize the ProbeBuilder directives in either webmethods.pbd or webmethods-toggles.pbd if you want to modify the default monitoring. For example, you can use the toggles file to fine-tune monitoring for specific components by turning on or turning off tracing for specific tracer groups.

- **webmethods.pbd**
  Monitors key services on the webMethods Integration Server, including flow services, Java services, trading networks, web services, XSLT services, and business processes.

- **webmethods-toggles.pbd**
  Toggles for turning on or turning off the monitoring of webMethods Integration Server components.

- **webmethods.pbl**
  Provides the default list of ProbeBuilder directive files for monitoring webMethods Integration Server:

  - webmethods.pbd

  - webmethods-toggles.pbd

Enable the Enterprise Manager Extension

CA APM for webMethods Integration Server files are installed by default in the <*EM_Home*> /examples/SOAExtensionForWebMethodsIS directory when you install the Enterprise Manager. To enable CA APM for webMethods Integration Server, you need to copy or move the Enterprise Manager files for webMethods Integration Server from the <EM_Home>/examples directory to the proper location in the Enterprise Manager home directory.

Note: CA APM for SOA must be enabled on the Enterprise Manager (https://docops.ca.com/display/APMDEVOPS101/Install+and+Configure+CA+APM+for+SOA#InstallandConfigureCAAPMforSOA-EnableExtensionsontheEnterpriseManager) before you can use CA APM for webMethods Integration Server.

Follow these steps:

1. Verify the CA APM for webMethods Integration Server directory, SOAExtensionForWebMethodsIS, is in the <*EM_Home*>/examples directory, then copy the files from the <*EM_Home*>/examples/SOAExtensionForWebMethodsIS directory to the corresponding location in the Enterprise Manager directory structure. For example, copy the files from the <*EM_Home*>/examples/SOAExtensionForWebMethodsIS/ext directory to the <EM_Home>/ext directory.

2. Remove the webMethods Integration Server Management Module, WebMethodsISManagementModule.jar, from the <*EM_Home*>/config/modules directory if the Enterprise Manager is a Collector in a clustered environment.
   You should only copy the Management Module to the <*EM_Home*>/config/modules directory on the Enterprise Manager you are using as the MOM computer. All other files and scripts should be installed on both the Collector Enterprise Managers and the MOM Enterprise Manager.

3. Restart the Workstation to load the dashboards and Overview tabs that are specific to CA APM for webMethods Integration Server.

Use Dashboards to Monitor webMethods

The SOA extension for webMethods Integration Server includes several preconfigured dashboards that you can use to monitor the overall health of the application environment. Dashboards aggregate data across deployed agents to summarize performance information and help you rapidly diagnose and resolve problems.

Typically, you use dashboards as the starting point for monitoring your environment because they let you do the following:

- Monitor the overall health, performance, availability, and current status of key components of webMethods Integration Server at-a-glance.
- Get early notification of potential problems in the production application environment when lower-level metrics signal that a caution or danger threshold has been crossed.
- Drill-down into performance information to isolate and identify which webMethods Integration Server business processes, services, or connection pools are experiencing delays or producing errors.

The preconfigured webMethods dashboards are packaged in the Enterprise Manager extension for webMethods as part of the webMethods Integration Server Management Module (*WebMethodsISManagementModule.jar*).

The webMethods Management Module provides the following preconfigured dashboards for webMethods Integration Server:

- **WebMethods - Home**
  A top-level architectural overview of WebMethods Integration Server and its key components, including alert indicators for the overall health of all services, business processes, trading network components, and external backend systems.
- **WebMethods - IS Services Overview**
  Summarized status for flow, Java, and XSLT services, and triggers, including graphs for Average Response Time, alert indicators for errors and stalls.
  From this dashboard, you can double-click the IS Slowest Services link or any Overall Health label to display the WebMethods - IS Slowest Services dashboard, which lists of the slowest flow, Java, and XSLT services, and a list of the slowest triggers.
- **WebMethods - Business Processes**
  Summarized status for all business processes, including graphs of the Average Response Time and Errors Per Interval at both the process and step level.
  The dashboard includes alert indicators for cancelled, restarted, suspended, and resumed operations at the process level; alert indicators for response time, stalls, concurrent invocations, and errors at the step level; and a list of the slowest business processes.

- **WebMethods - WebServices Overview**
  Summarized status for all client- and server side web services, including graphs of the Average Response Time to highlight client-side and server-side performance, lists of the slowest client-side and server-side services, and alert indicators for response time, SOAP faults, errors, and stalls for client and server services.
  From this dashboard, you can double-click the Web Service Operations link or any Overall Health label or graph to display the WebMethods - Web Service Operations dashboard, which includes graphs of the Average Response Time for client-side and server-side operations, alert indicators for SOAP faults and stalls, and lists of the slowest client-side and server-side web service operations.

- **WebMethods - Adapters**
  Summarized status for all webMethods adapters, including graphs of the Average Response Time for all adapter services, adapter notifications, and external backend systems.
  For adapter services, the dashboard also displays:

  - graphs for Errors Per Interval and Stall Count
  - alert indicators for concurrent invocations, errors, and stalls
  - a list of the slowest adapter services

  For adapter notifications and external backends, the dashboard also displays alert indicators for errors and stalls.

- **WebMethods - Trading Networks**
  Summarized status for all document processing handled by Trading Networks, including graphs of the Average Response Time and Errors Per Interval for all document types and processing rules, alert indicators for errors and stalls, and a list of the slowest documents processed.

- **WebMethods - Connection and Thread Pools**
  Summarized status for adapter connection pools, JDBC connection pools, and thread pools, including graphs for the number of available connections and the current pool size for adapter connections and JDBC connections, and graphs for the number of threads in use and the maximum number of threads in the thread pool.

You can view the preconfigured dashboards using the Workstation Console. You can also extend the webMethods Integration Server Management Module to include custom dashboards or modify the default dashboard definitions to include custom metrics or alerts.

Follow these steps:

1. Start the Enterprise Manager if it is not currently running.
2. Start the Workstation and log in to the Enterprise Manager where the SOA extension for webMethods is installed.
3. Click Workstation > New Console.
4. Select one of the webMethods dashboards from the Dashboard drop-down list. For example, select the WebMethods Home dashboard to see an overview of webMethods key components and internal workflow.

5. Double-click another tab or an alert in the dashboard to open the related dashboard to view more detailed information.For example, double-click a Java Services alert to see more detailed information about the overall health of webMethods Integration Server flow, Java, and XSLT services in the WebMethods - IS Services Overview dashboard:

6. From the WebMethods - IS Services Overview dashboard, you can double-click IS Slowest Services to display lists of the slowest individual flow, Java, XSLT, and trigger services.

7. Double-click a specific service, business process, or document metric in a dashboard to open the Investigator for further analysis.

8. From the WebMethods - IS Slowest Services dashboard, for example, you can double-click the slowest flow service to open the Investigator with that service's Average Response Time selected.

Filtering the services monitored and displayed

For the webMethods Integration Server, you can control which flow or Java services are monitored and included in the Investigator tree by specifying filters in a configuration file. The default configuration file is *wmExtension.config*. By editing this file, you can create filters using regular expressions that identify the flow services you want to include or exclude from monitoring.

About the Default Configuration File

The default configuration file, wmExtension.config, is located in the <*Agent_Home*>/common directory and is configured with default include and exclude filters. For example, the following default filter is defined to exclude webMethods built-in services:

```
com.wily.wm.service.filter.exclude=wm.*,pub.*
```

The configuration file also provides the following default filter to include matching webMethods services:

```
com.wily.wm.service.include=wm.tn:receive,wm.tn.route:routeBizdoc,pub.prt.tn:
handleBiZDoc
```

You can modify the default configuration file to include or exclude additional services, as needed, or remove the default filters if you want to monitor all webMethods services, including the built-in services. If no filters are defined in the wmExtension.config file, all services, including built-in services, are displayed in the Investigator tree.

Excluding services using regular expressions

Within the wmExtension.config configuration file, you can use the *com.wily.wm.service.filter.exclude* property to identify the services you want to exclude. For example, you can set the com.wily.wm. service.filter.exclude property with a regular expression that defines the service you want to exclude from monitoring. The exclude property can have any expression applicable for the fully-qualified name of the service.

For example, to exclude all flow services that end with the string *webservice*, set the exclude property like this:

```
com.wily.wm.service.filter.exclude=.*webservice
```

To define multiple filters, use a comma to separate the regular expressions. For example, to exclude all flow and Java services that start with *wm.server* and *wm.tomcat*, you can set the *com.wily.wm. service.filter.exclude* property like this:

```
com.wily.wm.service.filter.exclude=wm.server.*,wm.tomcat.*
```

All flow and Java services matching the regular expression are excluded and only the remaining services are displayed in the Investigator. If you specify a regular expression that is not valid, however, no services are excluded from monitoring and all flow and Java services, including built-in services, are displayed.

Specifying a Different Location for the Configuration File

Although the default wmExtension.config file is located in the <*Agent_Home*>/common directory, you can specify another directory or a different configuration file by specifying a different location in the start-up script for the server. For example, if an alternate wmExtension.config file is located in the directory C:\CA-Introscope, you can add the com.wily.wm.service.filter.fileloc property to the Java arguments in the start-up script for the server:

```
set JAVA_OPTS=%JAVA_OPTS% -Dcom.wily.wm.service.filter.fileloc=C:\CA-Introscope
```

If a file named wmExtension.config exists in the specified path, the agent checks for the include and exclude properties in that configuration file to determine which flow and Java services to monitor.

If the path to the configuration file is specified in the Java arguments of the server, the agent uses the filters specified in that file to determine the flow and Java services to exclude. If the path to the configuration file is not specified, the agent uses the filters defined in the <*Agent_Home*>/common /wmExtension.config file. If the <*Agent_Home*>/common/wmExtension.config or alternate wmExtension.config file is not found, no filtering is done and all flow and Java services, including built-in services, are displayed.

View and Navigate Metrics for webMethods

When you navigate in the Investigator tree, you can view the standard CA Introscope; metrics for most components of the webMethods Integration Server infrastructure. Data for the standard metrics is collected and aggregated into webMethods-specific metric categories displayed as nodes and sub-nodes in the Investigator tree. The specific metric categories and node names displayed depend on the processes, services, and resources you have deployed in your environment.

As you navigate through the Investigator tree, you can choose to view low-level metrics for individual operations or aggregated metrics depending on the node you select, enabling you to monitor the overall health of various services you have deployed through the webMethods Integration Server.

To view and navigate webMethods metrics in the Investigator

1. Expand the agent node, then click the WebMethods node to display the Overview tab, which lists summary information with the standard CA Introscope; metrics for all of the webMethods flow services and business processes you are monitoring.

2. Select a flow service or business process in the list to view all the standard metrics for that service or process in a graphical format.

3. Expand the WebMethods node to display sub-nodes for the top-level metric categories for webMethods Integration Server.

4. Click or expand a sub-node to display an Overview tab with summary information about that metric category. For example, click the Java Services node to display summarized metrics for Java services on the Overview tab.

5. Expand any sub-node to see more detailed information about individual business processes, flow services, Java services, or connection pools and the metrics associated with each. For example, you can expand the Flow Services node, then a specific flow service name and subfolder to display the metrics for an individual flow service.

Metrics for Adapters

Default webMethods adapters provide seamless connectivity to information resources and enterprise applications that help organizations achieve real-time execution of business processes. Adapters enable organizations to use webMethods to connect to packaged applications, such as SAP and Oracle application suites, or databases, such as Microsoft SQL Server and Oracle RDBMS, with minimal custom programming or time-consuming integration development. The specific adapters available for you to deploy depend on the version of webMethods Integration Server you are using.

The metrics for adapters provide detailed information about the specific adapters you have deployed and enable you to monitor the connections, operations, and events associated with those adapters.

You can monitor adapters using the metrics in the following metric categories under the WebMethods node:

- **Adapter Connection Pools**
  Adapter connections enable the Integration Server to connect to the external application or information store at run-time.

- **Adapter Notifications**
  Adapter notifications allow you to monitor the external resource and notify the Integration Server when an event initiated by the resource occurs. An adapter notification publishes a document to the webMethods Broker when the event occurs.

- **Adapter Services**
  Adapter services allow you to use the adapter's connection to the external resource to initiate an operation on the resource from the Integration Server.

For example, webMethods provides a JDBC Adapter to deliver a set of user interfaces, services, and templates that enable you to create integrations with databases using a JDBC driver. You can monitor the connections to the database using the metrics in the Adapter Connection Pools category.

JDBC Adapter services enable the Integration Server to initiate and perform database operations, such as insert, update, or delete data, on the database. Adapter notifications enable you to monitor the database and notify the Integration Server when an update, insert, or delete has occurred on a particular database table.

For example, an adapter service could enable a trading partner to query your inventory database to determine whether a particular item is currently in stock. And an adapter notification could notify the Integration Server when an update was performed on inventory database table.

To view and navigate adapter-related metrics in the Investigator:

1. Expand the agent node, then expand webMethods to display the adapter metric categories.

2. Expand WebMethods > Adapter Connection Pools, then expand a specific adapter name to view connection information for that adapter.

3. Expand WebMethods > Adapter Notifications, then expand a specific adapter name to view notification information for that adapter. If you have not configured any notifications for adapters, this metric category is not displayed.

4. Expand WebMethods > Adapter Services, then expand a specific adapter type to see the list of active connections.

5. Expand an active adapter connection to see the list of adapter services running on the Integration Server.

6. Expand an individual service to see the metrics for the service.

Metrics for Adapter Connection Pools

The following metrics are available for the adapters you have configured under the WebMethods > Adapter Connection Pools sub-nodes for individual adapter names:

- **Available Connections**
  Number of free connections available for the adapter.
  The available connections is calculated by subtracting the number of busy connections from the number of currently active connections. As the server tries to connect to the resource multiple times, the number of available connections decreases.

- **Current Size**
  Total number of currently active connections.

- **Maximum Size**
  Maximum number of connections allowed for the selected adapter.

- **Minimum Size**
  Minimum number of connections configured for the selected adapter.

Metrics for Adapter Notifications

If you have configured adapter notifications, a polling process or a listener process monitors the external resources for changes, such as an insert, update, or delete operation on a database table, so that the appropriate flow or Java services can react to the change.

For example, if you have deployed an adapter service to enable a trading partner to query the inventory database, you can configure an adapter notification to notify the Integration Server anytime an update is performed on inventory database table. To process a document associated with the notification, for example, to send an invoice based on the adapter notification, you can configure an Integration Server trigger.

All of the standard CA Introscope; metrics are available for adapter notifications under the WebMethods > Adapter Notifications sub-nodes for individual adapter service names.

Metrics for Adapter Services

All of the standard CA Introscope; metrics are available for adapter services under the WebMethods > Adapter Services sub-nodes for individual adapter connection service names.

The Errors Per Interval metric only includes errors that occur when the adapter service is executed. It does not include AccessExceptions or any other type of errors that occur before the service is executed.

Metrics for Authorization

The webMethods Integration Server typically includes multiple user groups with specific Access Control Lists (ACLs) that define the permissions different users are granted. For example, there are default permissions assigned to groups such as Administrators, Developers, Monitor Users, and Replicators. You can also create your own user groups and permissions, as needed. If users try to execute Integration Server services they do not have permission to access, they are denied access and an error is recorded.

You can view information about authorization failures by selecting the Errors Per Interval metric under the WebMethods > Authorization node.

Because the service is not invoked when the user is denied access, the error is not recorded in the service's Errors Per Interval metric. However, you can also view Access Exception errors by clicking the Errors tab in the Investigator. Select the error in the list to display detailed information about the error in the Error Snapshot.

Metrics for Business Processes

Business processes consist of a series of steps that complete a business event, such as placing an order or adding a new employee.

Only the following standard CA Introscope; metrics are available for webMethods business processes and business process steps under WebMethods > BusinessProcesses > <*business_process_name*> nodes:

- Average Response Time (ms)
- Errors Per Interval
- Responses Per Interval

At the business process level, these metrics track the time the process takes to complete successfully, the number of processes that completed successfully, and the number of processes that generated errors and failed. For distributed processes, the Average Response Time and Responses Per Interval at the process level are displayed only for the agent where the last step of the business process is executed.

In addition to the standard metrics, the following metrics are available for business processes:

- **Cancels Per Interval**
  Number of processes cancelled in the interval.
- **Suspends Per Interval**
  Number of processes suspended in the interval.
- **Restarts Per Interval**
  Number of processes that restarted in the interval.
- **Resumes Per Interval**
  Number of processes that resumed in the interval.

You can also collect dependency and deviation metrics for webMethods Integration Server business processes. For information about the standard dependency and deviation metrics, see Using Investigator to view SOA performance metrics (https://docops.ca.com/display/APMDEVOPS101/Monitor+a+Service-Oriented+Architecture#MonitoraService-OrientedArchitecture-UsingInvestigatortoviewSOAperformancemetrics).

View Step-Level Metrics for Business Processes

By default when you are monitoring webMethods Integration Server, all the standard CA Introscope; metrics are also available for business process steps as follows:

WebMethods > BusinessProcesses > <*business_process_name*> > <*step_identifier*> nodes

Valid for webMethods Integration Server 6.5.2 through 6.5.3: Step-level metrics are only displayed for single-level steps. Other step-level metrics, for example, metrics for human task steps that are executed internally as more than one step, referenced process, subprocess, or flow service steps, are not aggregated by default.

To get full step-level metrics for business processes when you are monitoring webMethods Integration Server versions 6.5.2 through 6.5.3, uncomment the following lines in the webmethods-toggles.pbd file:

```
TurnOn: BProc<version_number>FlowStepMarkTracing
TurnOn: BProc<version_number>StepFlowTracing
```

Follow these steps:

1. Expand the agent node, then expand webMethods > Business Processes.
   The business process that you have defined in the webMethods Designer and deployed as packages in your environment display.
   To differentiate between the different versions of business processes running, an underscore (_) and a version number are appended to the business process name.

2. Expand any process name.
   The steps that you have defined for that process appear.

3. Expand any step identifier (StepID).
   The step-level metrics and flow services metrics that represent the step on the Integration Server appear.

About Average Response Time for completed processes

It is possible for webMethods business processes to be canceled or suspended during execution from My WebMethods Server. Processes that are canceled or suspended using My WebMethods Server can also, in many cases, be resumed manually from My WebMethods Server. Because these process can be resubmitted from My WebMethods Server and run to completion, the Average Response Time metric at the process level only represents processes that complete successfully.

If a process fails or is suspended and is resubmitted from within My WebMethods Server, the Average Response Time metric reflects the time between the initial invocation of the process and its successful completion after being resubmitted. If a process fails and is not resubmitted using My WebMethods Server, it is not included in the Average Response Time metric. However, if the process is cancelled as part of its process flow, for example, using a Terminate step, it is included in the Average Response Time metric.

If more than one Integration Server is involved in the business process, the Average Response Time metric is only reported for the Integration Server where the business process finishes.

About Responses Per Interval for completed processes

The Responses Per Interval metric reflects the number of processes that completed successfully. If more than one Integration Server is involved in the business process, the Responses Per Interval metric is reported only for the Integration Server where the business process finishes.

About Errors Per Interval for failed processes

The Errors Per Interval metric is only shown when a process fails to run to completion.

About Concurrent Invocations for steps in a process

Some types of steps, such as SubProcess, Referenced Process, and Human Task steps, are executed internally as more than one step even though they are presented as a single step in the webMethods Designer. For example, Human Task steps are executed internally as two steps, *PRE_StepID* and *POST_StepID*. The Concurrent Invocations metric counts these internal steps as individual invocations rather than aggregating them into a single step.

Metrics for Flow Services

Flow Services are services written in webMethods Flow language. A flow service consists of a series of flow steps, each with clearly defined input and output. Each flow step is a basic unit of work that webMethods interprets and executes at run time. All the flow steps in a particular service are executed using the same thread so that the output of one step is available as input to the next step in the flow.

All of the standard CA Introscope; metrics are available for individual webMethods flow services under the WebMethods > Flow Services node. Data is collected for individual flow steps and aggregated into metrics for a flow service and across flow services to monitor the overall health of all flow services on the Integration Server.

The node names displayed are the fully-qualified names of the services you have chosen to monitor that have not been excluded using filters. For information about filtering the flow services to you are interested in monitoring, see Filtering the services monitored and displayed (see page 17).

Flow Services That Call Other Flow Services

In most cases, all of the standard CA Introscope; metrics and their corresponding aggregated metric values apply to webMethods flow services in the same way they apply to other application components.

If one flow service calls another flow service, metrics for both of these flow services are displayed as separate nodes under the WebMethods > Flow Services node in the Investigator tree. They are not nested with one node under the other.

To see all of the flow steps executed on the same webMethods Integration Server in calling sequence, you can start a transaction trace session.

Metrics for Java Services

Java services can be any user-defined or internal, built-in webMethods services that are implemented using the Java language, or written in other languages and wrapped using a Java class, and exposed as services on the webMethods Integration Server. For example, all of the built-in services that webMethods Integration Server provides are Java services.

All of the standard CA Introscope; metrics are available for the individual webMethods Java services you have deployed under the WebMethods > Java Services node.

In webMethods, Java services that reside in the same folder are methods of the same class. For example, a Java service with the fully-qualified name of *recording.user.accounts:createAccount* consists of the Java package (*recording.user*), Java class (*accounts*), and Java method (*createAccount*). The node names displayed in the Investigator represent the fully-qualified names of the services you have chosen to monitor and not excluded using filters.

For information about filtering the Java services to you are interested in monitoring, see Filtering the services monitored and displayed (see page 17).

Metrics for JDBC Pools

The webMethods Integration Server collects and stores information about users, documents, internal server functions, audit and error logs, and other information by connecting to databases through JDBC connection points. The maximum number of JDBC connections allowed in the JDBC pool can be configured using the Integration Server Administrator.

The following metrics are available for monitoring JDBC usage under the WebMethods > JDBC Pools sub-nodes for individual thread pool names:

- **Available Connections**
  Number of available connections for the JDBC pool.
  The available connections is calculated by subtracting the number of busy connections from the number of currently active connections.

- **Current Size**
  Total number of currently active JDBC connections.

- **Maximum Size**
  Maximum number of JDBC connections configured for the selected pool.

- **Minimum Size**
  Minimum number of JDBC connections configured for the selected pool.

Metrics for Thread Pools

The webMethods Integration Server uses threads to execute services, retrieve documents from the webMethods Broker, and execute triggers. When the server starts, the thread pool initially contains the minimum number of threads. The server adds threads to the pool as needed until it reaches the maximum allowed. If the maximum number of threads are in use, the server waits until processes complete, then returns threads to the pool before beginning more processes.

The following metrics are available for monitoring thread usage under the WebMethods > Thread Pools sub-nodes for individual pool names:

- **Maximum Size**
  Maximum number of threads configured for the selected thread pool.
- **Minimum Size**
  Minimum number of threads configured for the selected thread pool.
- **Used Threads**
  Number of threads currently in use for the selected thread pool.

Metrics for Trading Networks

Trading networks enable organizations to exchange documents to establish and enrich business-to-business relationships. For example, you can identify buyers, suppliers, strategic partners or other organizations as trading partners with which you exchange documents. By exchanging documents, you can streamline business processes that cross organizational boundaries. With webMethods Integration Server, trading networks act as the gateway between trading partners.

The following standard metric is available for webMethods trading networks under the WebMethods > Trading Networks node:

- Errors Per Interval

You can also monitor trading networks for XML document recognition and processing rules with the following metric categories under the WebMethods > Trading Networks node:

- **Document Types**
  Document types identify the structure and type of data to be exchanged in different partner relationships.
- **Processing Rules**
  Processing rules describe how a document is routed through the Integration Server and to its destination.
- **Service Execution Tasks**
  Service execution tasks are created when a processing rule executes the service asynchronously with a retry limit.

- **Service Threads**
  Service threads are created to handle asynchronous processing for documents being routed without a retry limit.

For example, you can expand the WebMethods > Trading Networks > Document Types sub-node to view metrics for specific document types, including document recognition and acceptance metrics for individual document type. Similarly, you can expand the WebMethods > Trading Networks > Processing Rules sub-node to view metrics for specific processing rules, including metrics for individual pre-routing and routing operations.

Metrics for Document Types

Only the following standard CA Introscope; metrics are available for webMethods trading networks under the WebMethods > Trading Networks > Document Types node:

- Average Response Time (ms)
- Responses Per Interval

The Average Response Time and Responses Per Interval metrics for Document Types are aggregated when documents are processed. These metrics are only reported when processing rules are executed. If a document is submitted, but not processed by any processing rules, the Average Response Time and Responses Per Interval metrics are not reported.

In addition to the standard metrics, the following metric is available under the WebMethods > Trading Networks > Document Types node when valid documents are submitted:

- **Recognitions Per Interval**
  Number of documents recognized as valid trading partner documents at the end of a 15-second interval.

For the individual document types recognized, all of the standard CA Introscope; metrics are available under the WebMethods > Trading Networks > Document Types > <*document_type_name*> node.

Metrics for Processing Rules

Only the following standard CA Introscope; metrics are available for webMethods trading networks under the WebMethods > Trading Networks > Processing Rules > <*processing_rule_name*> sub-nodes for individual processing rules:

- Average Response Time (ms)
- Responses Per Interval

All of the standard CA Introscope; metrics are available for webMethods individual processing rule operations under the WebMethods > Trading Networks > Processing Rules > <*processing_rule_name* > > PreRoute or Route Actions sub-nodes for the specific operations in a processing rule.

About Average Response Time for Processing Rules

The Average Response Time metric for a processing rule aggregates the average response time taken for routing the document synchronously. For synchronous invocations in a processing rule, the metric is the same as the aggregated Average Response Time for Document Types.

Metrics for Service Execution Tasks

The following metrics are available for webMethods trading networks under the WebMethods > Trading Networks > Service Execution Tasks sub-node:

- New Tasks Per Interval
- Task Failures Per Interval
- Tasks Completed Per Interval

In addition, the following standard CA Introscope; metrics are available for individual invocation operations under the WebMethods > Trading Networks > Service Execution Tasks > invoke sub-node:

- Average Response Time (ms)
- Responses Per Interval

Metrics for Service Threads

The following standard CA Introscope; metrics are available for webMethods trading networks under the WebMethods > Trading Networks > Service Threads sub-node:

- Average Response Time (ms)
- Responses Per Interval

Metrics for Triggers

You can configure Integration Server triggers to use webMethods Broker or the Java Message Service (JMS) to process documents. A webMethods Broker trigger is a trigger that subscribes to and processes documents that are published locally or delivered to the webMethods Broker. A JMS trigger is a trigger that receives messages from a destination, for example, a queue or topic, on a JMS provider then processes those messages.

All of the standard CA Introscope; metrics are available for triggers under the WebMethods > Triggers sub-nodes for individual trigger names. If you have not configured any Broker, local, or JMS triggers, this metric category is not displayed.

Metrics for WebServices

Web services are building blocks that are packaged as a unit and published to a network so they are available to users or software programs.

- Valid for a supported version of webMethods Integration Server -- The web service connector defines whether the Integration Server is acting as a web service *consumer* (client) or as a web service *provider* (server). For example, you can expose any flow service or Java-based service you deploy externally as a web service. You use a provider web service descriptor that publishes information about the service to a UDDI registry. The webMethods Integration Server can also invoke web services on external application servers using a consumer web service descriptor to request service as a client.

- Valid for webMethods Integration Server 6.5.2 through 6.5.3 -- You can generate web service *connectors* from WSDL documents to invoke remote web services. A web service connector is a flow service that has an input and output signature. The signature corresponds to the input and output messages from the WSDL document from which it was created. With webMethods Integration Server, you can also use Integration Server and Developer to turn any existing service in an Integration Server package into a web service.

All the standard CA Introscope; metrics are available for client and server web services and operations on the webMethods Integration Server. In addition, the SOAP Faults Per Interval is a standard metric for all extensions that monitor web services on SOA platforms.

Note: For webMethods Integration Server requirements, see the *Compatibility Guide (https://wiki.ca.com/display/APMDEVOPS98/Component+Version+Compatibility)*.

About server-side metrics

As a web service provider, webMethods has multiple SOAP processors to handle web service requests:

- the default web services SOAP processor

- the SOAP RPC processor to receive and process SOAP remote procedure calls

- the default SOAP message handler to process messages when a process directive is undefined or omitted

The server-side web service metrics include information for all three SOAP processors. Custom SOAP processors are not included in the metrics, however.

About client-side metrics

Client side metrics represent the execution of a web service request running on external application server.

When the webMethods Integration Server acts as a web service consumer, it automatically generates a web service *connector* for each operation. The client then binds directly to the endpoint of the web service through the connector. When the web service connector executes, the request to invoke the web service goes directly to the web service implementation. Internally, web service connectors are flow services and can be monitored using the Flow Services metrics.

The WebServices > Client metrics represent the execution of the connector calling the external web service and not operations the client is requesting directly.

Server side metrics represent the execution of the web service running on the webMethods Integration Server.

Metrics for XSLT Services

The Extensible Stylesheet Language (XSL) and XSL Transformations (XSLT) provide an XML-based language for transforming source XML documents into other documents. For example, the original XML document can be used to create a new XML document, converted into HTML for display as a Web page, published as plain text.

When you invoke an XSLT service, the Integration Server retrieves the instructions in an associated document, the stylesheet, then applies those instructions to transform the source XML document into a new document in the format defined by the stylesheet.

All of the standard CA Introscope; metrics are available for monitoring individual webMethods XSLT services under the WebMethods > XSLT Services node.

Viewing default webMethods metric groupings

The SOA extension for webMethods Integration Server includes default metric groupings that are used to define the default dashboards and alerts. You can also use these default metric groupings in custom dashboards and alerts.

The default metric groupings are packaged in the Enterprise Manager extension for webMethods Integration Server as part of the webMethods Integration Server Management Module (*WebMethodsISManagementModule.jar*).

To view the default metric groupings for webMethods agents

1. In the Investigator, click Workstation > New Management Module Editor.
2. Expand *SuperDomain* > Management Modules > WebMethods IS (*Super Domain*).
3. Expand the Metric Groupings node to view all of the metric groupings defined for the webMethods management module.
4. Click a specific metric grouping to view its definition in the Viewer pane.
   You can modify the default settings for any metric grouping or create your own custom metric groupings

Viewing default webMethods alerts

The SOA extension for webMethods includes default alert definitions that are used in the preconfigured dashboards. You can also use these default alerts in custom dashboards. Most of the default alerts are preconfigured with default Caution and Danger thresholds and to send notification to the console if a threshold is crossed or severity increases.

The default alert definitions are packaged in the Enterprise Manager extension for webMethods Integration Server as part of the webMethods Integration Server Management Module (*WebMethodsISManagementModule.jar*).

To view the default alert definitions for webMethods agents:

1. In the Investigator, click Workstation > New Management Module Editor.
2. Expand *SuperDomain* > Management Modules > WebMethods IS (*Super Domain*).
3. Expand the Alerts node to view all of the alerts defined for the webMethods Integration Server management module.
4. Click a specific alert to view its definition in the Viewer pane.

In particular, you should review the default Caution and Danger thresholds and predefined actions for critical alerts and tune them for your environment. For example, you may want adjust the threshold values, if necessary, add notifications, or define corrective actions.

You can modify any of the default settings for any alert or create your own custom alerts.

Note: For more information about creating or modifying alerts, see the *CA APM Workstation User Guide*.

Viewing webMethods dependencies

You can view dependencies for webMethods flow, java, adapter, and web services and webMethods business processes by selecting an appropriate webMethods node in the Investigator tree and clicking the SOA Dependency Map tab.

The node you select determines the context displayed in the dependency map. You can then roll up to collapse or roll down to expand the context and level of detail you are viewing. For example, for a top-level view of dependencies for all business processes, you can select the Business Process node in the Investigator and click the SOA Dependency Map tab:

To see a high level view of dependencies for a specific business process, you can select the business process name in the Investigator and continue to add dependency levels to the map to see the entire workflow for the business process or zoom in on specific nodes of the map, as needed. For more detailed information about navigating the dependency map, see Using the SOA Dependency Map (https://docops.ca.com/display/APMDEVOPS101/Using+the+SOA+Dependency+Map).

Tracing transactions for webMethods

Transaction tracing provides a detailed or summary view of the specific steps involved in completing a business transaction. For webMethods Integration Server business processes or application services, you can trace transactions that include operations routed through the following protocols:

- Simple Object Access Protocol (SOAP)
- Hypertext Transport Protocol (HTTP)
- Hypertext Transport Protocol Secure (HTTPS)
- Java Message Service (JMS)
- webMethods Broker Message Service

Transactions that involve webMethods services can include synchronous and asynchronous calls and activities running in parallel using different threads. To enable transaction tracing for transactions that span multiple threads or processes, correlation identifiers are inserted and consumed as a transaction steps through each component and operation. You can then view detailed information about the specific operations performed and the time each operation took to complete.

You can also track a business transaction across any combination of platforms as long as CA APM for SOA and CA APM for webMethods Integration Server are enabled at every node being traced. This enables you to view details about the transaction even if the transaction spans multiple JVMs or CLRs.

About Configuring Cross-Process Transaction Tracing

Typically, you can configure the cross-process transaction tracing feature by enabling correlation information to be passed from one process to another. Information is passed using either SOAP or HTTP headers. Valid for webMethods Integration Server version 6.5.2 through 6.5.3 -- you cannot use SOAP headers for passing correlation identifiers when you are using RPC style web service clients (Connectors).

To see transaction traces for business transactions involving webMethods Integration Server RPC Web Service Connectors, configure agents correlated tracing (https://docops.ca.com/display/APMDEVOPS101/Configure+SOA-Specific+Properties#ConfigureSOA-SpecificProperties-Configuringcorrelatedtracing) to use HTTP headers for passing correlation identifiers.

Understanding the value of cross-process transaction tracing

Cross-process transaction tracing provides valuable information about the operations being performed by loosely coupled services in a service-oriented architecture. You can use cross-process transaction tracing to determine:

- how business processes are routed through flow or Java services.
- which services are called and executed during a transaction.
- the sequence of calls made during a transaction.
- where the processing of a request or reply is slowest.

Starting and viewing a sample transaction trace

You can start a transaction trace session in the following ways:

- Directly from a map node in the SOA Dependency Map.
- Manually from the Workstation by clicking Workstation > New Transaction Trace Session.

If you start the transaction trace from the dependency map, the map node type sets the default filter automatically. If you manually start a new transaction trace session, you can select one of the following filter types for webMethods:

- businessprocess
- namespace
- operationname

After you configure filters and start the transaction trace session, the Transaction Trace Viewer is displayed. You can select a trace to view additional details about the calls made in the transaction. These details include any triggers, flow service operations, or business process steps executed on a webMethods Integration Server.

Using the Sequence View for webMethods processes

Because transactions involving webMethods Integration Server often include asynchronous calls, you may find it useful to click the Sequence View to view the transaction workflow for the processes that executed asynchronously as part of a transaction. The Sequence View displays the order in which

process execute to the extent the sequence can be identified. For webMethods Integration Server transactions, the sequence does not necessarily represent a traditional caller-callee relationship, but illustrates when one process triggers the execution of another process.

You should note, however, that the processing time for webMethods Integration Server processes is calculated using the full duration from start to completion of the process, which includes the processing time associated with its called processes. Net duration, which subtracts the processing time for non-blocking synchronous and asynchronous processes from the calling process's duration, is not supported for webMethods Integration Server processes.

For more information about transaction tracing, see Using transaction tracing in SOA environments (https://docops.ca.com/display/APMDEVOPS101/Use+Transaction+Tracing+in+SOA+Environments).For information about configuring tracing, see the *CA APM Java Agent Implementation Guide* or the *CA APM .NET Agent Implementation Guide*. For more information about working with transaction trace views and historical data, see the *CA APM Workstation User Guide*.