



FuseSource

A Progress Software Company

Fuse HQ

Fuse Mediation Router Metrics Reference

Version 4.4
April 2011

The experts in open source integration and messaging

Fuse Mediation Router Metrics Reference

Version 4.4

Updated: 03 Aug 2011

Copyright © 2011 FuseSource Corporation, a Progress Software company. All rights reserved.

Legal Notices

These materials and all Progress© software products are copyrighted and all rights are reserved by Progress Software Corporation. The information in these materials is subject to change without notice, and Progress Software Corporation assumes no responsibility for any errors that may appear therein. The references in these materials to specific platforms supported are subject to change.

Actional, Apama, Apama (and Design), Artix, Business Empowerment, DataDirect (and design), DataDirect Connect, DataDirect Connect64, DataDirect Technologies, DataDirect XML Converters, DataDirect XQuery, DataXtend, Dynamic Routing Architecture, EdgeXtend, Empowerment Center, Fathom, Fuse Mediation Router, Fuse Message Broker, Fuse Services Framework, IntelliStream, IONA, IONA (and design), Making Software Work Together, Mindreef, ObjectStore, OpenEdge, Orbix, PeerDirect, POSSENET, Powered by Progress, PowerTier, Progress, Progress DataXtend, Progress Dynamics, Progress Business Empowerment, Progress Empowerment Center, Progress Empowerment Program, Progress OpenEdge, Progress Profiles, Progress Results, Progress Software Developers Network, Progress Sonic, ProVision, PS Select, Savvion, SequeLink, Shadow, SOAPscope, SOAPStation, Sonic, Sonic ESB, SonicMQ, Sonic Orchestration Server, SonicSynergy, SpeedScript, Stylus Studio, Technical Empowerment, WebSpeed, Xcalia (and design), and Your Software, Our Technology—Experience the Connection are registered trademarks of Progress Software Corporation or one of its affiliates or subsidiaries in the U.S. and/or other countries. AccelEvent, Apama Dashboard Studio, Apama Event Manager, Apama Event Modeler, Apama Event Store, Apama Risk Firewall, AppsAlive, AppServer, ASPen, ASP-in-a-Box, BusinessEdge, Business Making Progress, Cache-Forward, DataDirect Spy, DataDirect SupportLink, Fuse, Future Proof, GVAC, High Performance Integration, ObjectStore Inspector, ObjectStore Performance Expert, OpenAccess, Orbacus, Pantero, POSSE, ProDataSet, Progress ESP Event Manager, Progress ESP Event Modeler, Progress Event Engine, Progress RFID, Progress Software Business Making Progress, PSE Pro, SectorAlliance, SeeThinkAct, Shadow z/Services, Shadow z/Direct, Shadow z/Events, Shadow z/Presentation, Shadow Studio, SmartBrowser, SmartComponent, SmartDataBrowser, SmartDataObjects, SmartDataView, SmartDialog, SmartFolder, SmartFrame, SmartObjects, SmartPanel, SmartQuery, SmartViewer, SmartWindow, Sonic Business Integration Suite, Sonic Process Manager, Sonic Collaboration Server, Sonic Continuous Availability Architecture, Sonic Database Service, Sonic Workbench, Sonic XML Server, The Brains Behind BAM, WebClient, and Who Makes Progress are trademarks or service marks of Progress Software Corporation and/or its subsidiaries or affiliates in the U.S. and other countries. Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. Any other trademarks contained herein are the property of their respective owners.

Table of Contents

1. Accessing and Using Collected Router Metrics	7
2. Fuse Mediation Router Routing Metrics	15
3. Fuse Mediation Router Processor Metrics	23

List of Figures

1.1. Setting the metric collection interval	8
1.2. Setting the metric refresh rate	9
1.3. Resource's Control tab	9
1.4. Control Action Properties and Schedule portlet	10
1.5. Control Action Schedule section	11
1.6. Resource's Alert tab	11
1.7. New Alert Definition portlet:Condition Set section	11
1.8. Alert Response tabs	12
1.9. Control Action portlet	12
2.1. route1	15
2.2. Availability Statistics Icons	16
3.1. bean1 Processor	23

Chapter 1. Accessing and Using Collected Router Metrics

Overview

Fuse Mediation Router metrics are collected for routes that are deployed in Fuse ESB. Based on a resource's metrics, you can invoke Control Actions on the resource to avoid or correct throughput and performance problems.

Fuse HQ Metric Collectors

The main service entries, which collect the required metrics, are listed in the server's and each agent's `fuse-routing-services.xml` file. This file is located in:

- On the server

```
./server-4.4.0.2-fuse/hq-engine/server/default/deploy/hq.ear/hq-plugins.1
```

- On each agent

```
./agent-4.4.0.2-fuse/bundles/agent-4.4.0.2-fuse/pdk/plugins.
```



Note

When a deployment consists of multiple agents, each agent's `fuse-routing-services.xml` file must remain in sync with the Server's `fuse-routing-services.xml` file so that data is collected properly. If, for example, you changed a property of a metric in the Server's `fuse-routing-services.xml` file, but not in the agent's, not only would the metric not display as expected, it might not even be collected.

The main service entries are:

- Router
- Router Component
- Router Endpoint
- Route

¹The path to the server's or the agents' `fuse-routing-services.xml` file is version-specific.

- Processors

Fuse HQ Data Collection Basics



Tip

For a background on how Fuse HQ collects and renders data, visit [Overview of Fuse HQ](#)² and [Fuse HQ Camel Metrics FAQ](#)³.

After resources are autodiscovered or manually added to Fuse HQ, data related to them is collected at configurable intervals. By default, performance data is collected every five minutes, while availability and throughput data are collected every ten minutes.

At the specified collection time, the Fuse HQ agent queries Fuse ESB, gathering statistics from the relevant services and sending the data to Fuse HQ server's database. At the specified **Metrics Refresh** rate, the Fuse HQ server retrieves and displays the new data.

For a selected resource, on the  portlet, you can:

- Reset the collection interval for individual metrics for a specified number of minutes, hours, days, or weeks.

Figure 1.1. Setting the metric collection interval



Note

For a grouped resource, if you reset the collection interval of selected metrics for only some of the member resources, the collection interval of those metrics then displays *VARIES*, instead of a time interval.

- Reset the rate at which Fuse HQ automatically refreshes the **Indicator** and **Metrics** tabs with updated data. The default is two minutes.

² <http://fusesource.com/wiki/display/HQDOC440/Overview+of+HQ>

³ <http://fusesource.com/wiki/display/HQDOC440/Fuse+HQ+Camel+Metrics+FAQ>

Figure 1.2. Setting the metric refresh rate

Metrics Refresh: 1 min | 2 min | 5 min | OFF

Controlling Resources

Each of the main services provides a defined set of Control Actions that enable you to control the selected resource and to retrieve its raw statistics data.

You can issue Control Actions from the selected resource's **Control** tab or by including them in an Alert defined for the resource.

Control Actions via the Control Tab

From a resource's **Control** tab, you can issue Control Actions immediately or schedule them at a later time for once-only or repeat-at-intervals operation.

- For example, to issue an immediate control action, on the resource's **Control** tab:

Figure 1.3. Resource's Control tab

MonitorInventoryAlertControlViews

CurrentHistory

Current Status

Control Action: getExchangesCompleted
Command State: ✓ Completed
Command Status: 3302
Elapsed Time: 240ms
[Clear Status Detail](#)

Quick Control - Quick Control Actions will occur after the current Control Action.

Control Action:

Select...

Control Arguments (optional):

Quick Control Actions will be done in parallel to all resources.

Control Action Schedule - Click "New..." below to schedule a Control Action.

<input type="checkbox"/> Control Action ▲	Next Fire	Date Scheduled
NEW...		DELETE

1. In the **Quick Control** section, select a Control Action from the drop-down menu.

2. Add any optional argument the Control Action supports.

See ["Route Control Actions" on page 18](#) and ["Processor Control Actions" on page 26](#) for supported arguments.

3. Click .


Fuse HQ displays results in the **Current Status** section, as shown in [Figure 1.3](#).

- Or, to schedule a control action at a later time:

1. At the bottom of the **Control** tab, click [NEW...](#) to open the **Control Action Properties and Schedule** portlet ([Figure 1.4](#)).

Figure 1.4. Control Action Properties and Schedule portlet



2. Select a Control Action.
3. Specify the date and time to activate it.
4. Select the interval (never, monthly, weekly, or daily) at which you want to repeat the Control Action.
5. click  to return to the resource's **Control** tab.

The **Control Action Schedule** section ([Figure 1.5](#)) displays the details about the scheduled Control Action.

Figure 1.5. Control Action Schedule section

Control Action Schedule - Click "New..." below to schedule a Control Action.		
<input type="checkbox"/> Control Action ▲	Next Fire	Date Scheduled
<input type="checkbox"/> getDescription	07/18/2011 05:45 PM	Once, on 7/18/11 @ 5:45 PM
<input type="button" value="NEW..."/> <input type="button" value="DELETE"/>		

Control Actions via Alert Definitions

Within a resource's Alert definition, you can enable a Control Action in response to a defined alert.

Figure 1.6. Resource's Alert tab

1. On the Resource's **Alert** tab (Figure 1.6), click the **Configure** subtab.
2. Click **NEW...** to open the **New Alert Definition** portlet (partially shown in Figure 1.7).

Figure 1.7. New Alert Definition portlet:Condition Set section

3. In the **Alert Properties** section (not shown), name and describe the alert, then select a priority and state (active/inactive).
4. In the **Condition Set** section, select the metric and set up the condition that will trigger the alert. Select how often to activate the Control Action and to generate alerts.
5. When done configuring the alert, click **Ok** to open a set of tabs (Figure 1.8) that spans the bottom of the **Condition Set** section.

Figure 1.8. Alert Response tabs

The screenshot shows the 'Condition Set' configuration window. At the top, it displays 'If Condition: ExchangesCompleted per Minute < 25.0' and 'Enable Action(s): Each time conditions are met.' Below this is an 'EDIT...' button. A horizontal tab bar contains five tabs: 'Escalation', 'Control Action' (which is highlighted with an orange border), 'Notify Roles', 'Notify HQ Users', and 'Notify Other Recipients'. Below the tabs, it shows 'Control Type: none' and another 'EDIT...' button.

6. Click **Control Action**, then click **EDIT...** to open the **Control Action** portlet (Figure 1.9).

Figure 1.9. Control Action portlet

The screenshot shows the 'Control Action' configuration portlet. It has three fields: 'Resource Type' with a dropdown menu showing 'ServiceMix 4.0-fuse Route', 'Resource Name' with a text field containing 'Jane-Murpheys-MacBook-Pro.local ServiceMix 4.0-fuse "route1" Jane-Murpheys-MacBook-Pro.local/217-camel-3 Route', and 'Control Type' with a dropdown menu showing 'getExchangesCompleted'. At the bottom right are three buttons: 'Ok' (green), 'Reset' (grey), and 'Cancel' (grey).

7. Select the Control Action from the **Control Type** drop-down menu.
8. Click **Ok**, then click **<< Return to Alert Definitions** to review the list of all defined alerts.

For a description of the supported route Control Actions, see ["Fuse Mediation Router Routing Metrics" on page 15](#) and ["Route Control Actions" on page 18](#).

For a description of the supported processor Control Actions, see ["Processor Control Actions"](#) on page 26.

Chapter 2. Fuse Mediation Router Routing Metrics

Overview

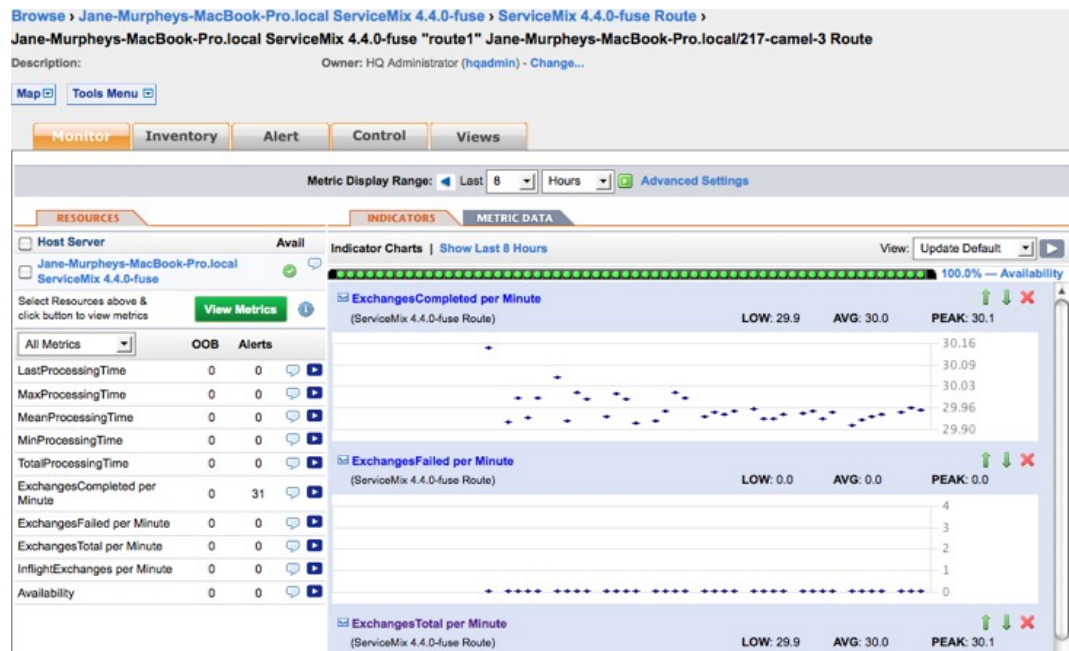
This chapter describes the route metrics collected by the Route service and supported control actions.

Route Metrics

The Route service collects metrics related to Camel routes that run inside CamelContexts. For example, route1 identifies the following route:

```
Jane-Murpheys-MacBook-Pro.local ServiceMix 4.4.0-fuse "route1"  
new-host-2.home/217-camel-3 Route
```

Figure 2.1. route1








Supported metrics are:

- `Availability`

This metric is of category *Availability* and of collection type¹ *Dynamic*.

Indicates the percentage of time a route was available during a given time slice.

Figure 2.2. Availability Statistics Icons

	100% availability
	0% availability
	Availability between 0% to 100%
	Resource is paused
	No availability was collected in the specified time frame

The following metrics are of category *Throughput* and collection type *Trends Up*:

- `ExchangesCompleted`

Indicates the total number of exchanges the route has processed successfully. By default, Fuse HQ displays the absolute number of exchanges that have completed the route since route start-up or the last `reset` operation.

- `ExchangesFailed`

Indicates the total number of exchanges that the route has failed to process. By default, Fuse HQ displays the absolute number of exchanges that have failed since route start-up or the last `reset` operation.

- `ExchangesTotal`

Indicates the total number of exchanges, passed or failed, that the route has processed. By default, Fuse HQ displays the absolute number of exchanges that have occurred since route start-up or the last `reset` operation.

¹On some screens, you may see the term *Value Type* instead of *Collection Type*.

- `InflightExchanges`

Indicates the number of exchanges currently transiting the route.

The following metrics are of category *Throughput* and collection type *Dynamic*.

Because Trends Up metrics continuously increase, the rate of change becomes more important to track. So Fuse HQ automatically calculates and creates a secondary *per Minute* metric for each Trends Up metric. By default, the *per Minute* Throughput metrics are enabled and charted.

- `ExchangesCompleted per Minute`

Indicates the number of exchanges the route has successfully processed per minute, since route start-up or the last `reset` operation.

- `ExchangesFailed per Minute`

Indicates the number of exchanges that the route has failed to process per minute, since route start-up or the last `reset` operation.

- `ExchangesTotal per Minute`

Indicates the number of exchanges, passed or failed, that the route has processed per minute, since route start-up or the last `reset` operation.

The following metrics are of category *Performance* and collection type *Dynamic*:

- `LastProcessingTime`

Indicates the time, in milliseconds, it took the route to process the last exchange.

- `MaxProcessingTime`

Indicates the longest time, in milliseconds, to process an exchange since route start-up or the last `reset` operation.

- `MeanProcesssingTime`

Indicates the average processing time, in milliseconds, for all exchanges processed since route start-up or the last `reset` operation.

- `MinProcessingTime`

Indicates the shortest time, in milliseconds, to process an exchange since route start-up or the last `reset` operation.

- `TotalProcessingTime`

Indicates the total processing time, in milliseconds, of all exchanges processed since route start-up or the last `reset` operation.

Route Control Actions

Route control actions enable you to control routes and to retrieve performance and statistics data from them.

Supported control actions are:

- `start`

The start operation starts up the selected route, including all of its processors and services.

- `stop`

The stop operation provides three ways to stop the selected route, including all of its processors and services.

- `stop()`—Stop the route immediately pending completion of all inflight exchanges.
- `stop(int timeout, abortAfterTimeout)`—Stop the route after specified `timeout` milliseconds, but if `abortAfterTimeout = true`, abort the stop operation and restart the route to complete exchanges still in flight.

(The default `timeout` is 300 seconds.)

- `stop(int timeout)`—Stop the route after the specified `timeout` milliseconds.
- `reset`



Warning

The reset operation immediately clears all counters, thus all statistics, for the selected route.

- `shutdown`

The shutdown operation provides two ways to shutdown the selected route, including all of its processors and services.

- `shutdown()`—Stop and shutdown the route immediately pending completion of all inflight exchanges.
- `shutdown(int timeout)`—Stop and shutdown the route after the specified `timeout` milliseconds.

- `getState`

The `getState` operation returns the state of the route:

- `started`
- `stopped`
- `suspended`
- `resumed`
- `shutdown`

- `getDescription`

The `getDescription` operation returns the description of the route; for example,

```
EventDrivenConsumerRoute[Endpoint[timer://springTimer?fixedRate=true&period=2000]
->
Instrumentation:route[UnitOfWork(Pipeline[ [Channel[BeanProcessor[bean:
myTransform]]],
Channel[sendTo(Endpoint[log://ExampleRouter]] )]] ]]
```

- `getCamelID`

The `getCamelID` operation returns the ID of the `CamelContext` in which the selected route is running; for example,

```
217-camel-3
```

- `setTracing`

The `setTracing` operation enables or disables tracing—the logging of exchanges that transit the selected route.

- `getRoutingPolicyList`

The `getRoutingPolicyList` operation returns a list of the active routing policies; for example,

- `ThrottlingInflightRoutePolicy`
 - `CronScheduledRoutePolicy`
 - `SimpleRoutePolicy`
- `getEndpointUri`

The `getEndpointUri` operation returns the selected route's endpoint URI; for example,

```
timer://springTimer?fixedRate=true&period=2000
```

- `getRouteId`

The `getRouteId` operation returns the selected route's ID; for example,

```
route1
```

- `getTracing`

The `getTracing` operation reports whether tracing—the logging of exchanges that transit the selected route—is enabled (`true`) or disabled (`false`).

- `getInflightExchanges`

The `getInflightExchanges` operation returns the number of exchanges the route is currently processing.

- `setStatisticsEnabled`

The `setStatisticsEnabled` operation enables (`true`) or disables (`false`) on the selected route.

- `getExchangesCompleted`

The `getExchangesCompleted` operation returns the number of exchanges the route has processed successfully.

- `getExchangesFailed`

The `getExchangesFailed` operation returns the number of exchanges the route has failed to process successfully.

- `getMinProcessingTime`

The `getMinProcessingTime` operation returns the minimum time, in milliseconds, taken to process an exchange since route start-up or the last `reset` operation.

`getMeanProcessingTime`

The `getMeanProcessingTime` operation returns the average time, in milliseconds, taken to process an exchange since route start-up or the last `reset` operation.

- `getMaxProcessingTime`

The `getMaxProcessingTime` operation returns the maximum time, in milliseconds, taken to process an exchange since route start-up or the last `reset` operation.

- `getTotalProcessingTime`

The `getTotalProcessingTime` operation returns the total time, in milliseconds, taken to process all exchanges, both failed and completed, since route start-up.

- `getLastProcessingTime`

The `getLastProcessingTime` operation returns the time, in milliseconds, taken to process the most recent exchange.

- `getLastExchangeCompletedTimestamp`

The `getLastExchangeCompletedTimestamp` operation returns the date and time of the last successfully processed exchange since route start-up or the last `reset` operation.

- `getFirstExchangeCompletedTimestamp`

The `getFirstExchangeCompletedTimestamp` operation returns the date and time of the first successfully processed exchange since route start-up or the last `reset` operation.

- `getLastExchangeFailureTimestamp`

The `getLastExchangeFailureTimestamp` operation returns the date and time of the last failed exchange since route start-up or the last `reset` operation.

- `getFirstExchangeFailureTimestamp`

The `getFirstExchangeFailureTimestamp` operation returns the date and time of the first failed exchange since route start-up or the last `reset` operation.

- `isStatisticsEnabled`

The `isStatisticsEnabled` operation reports whether statistics is enabled (`true`) or disabled (`false`) on the selected route.

- `getExchangesTotal`

The `getExchangesTotal` operation returns the total number of exchanges, both failed and completed, the selected route has processed since route start-up or the last `reset` operation.

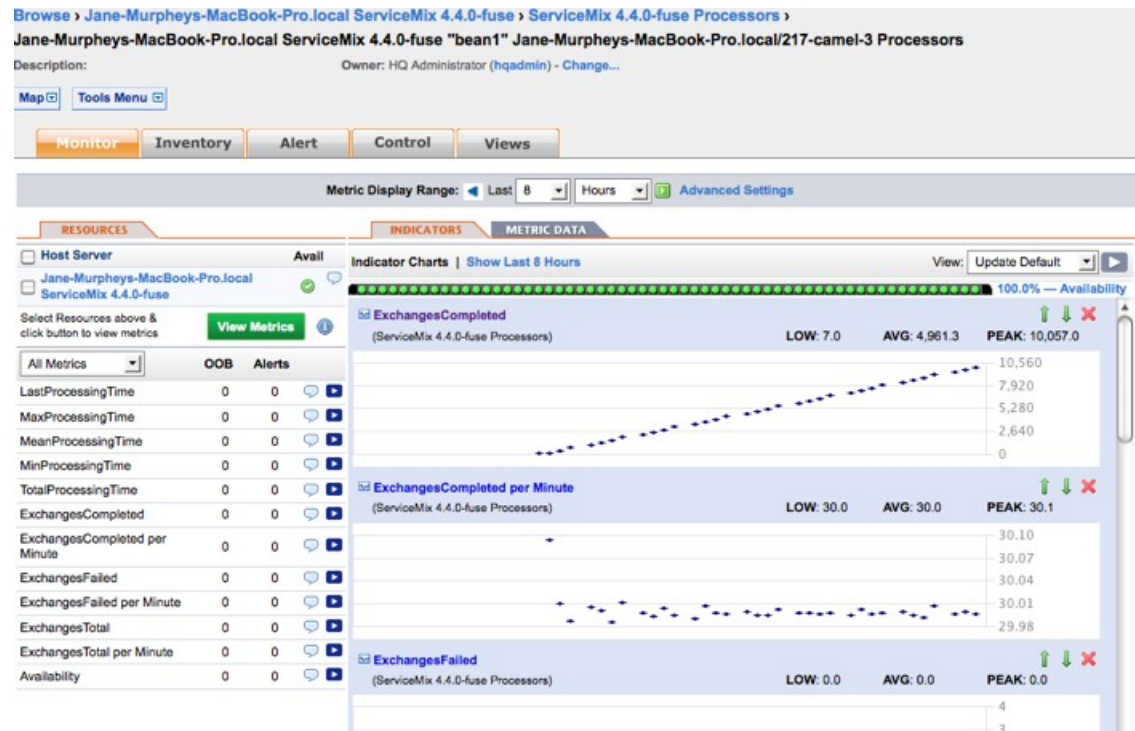
Chapter 3. Fuse Mediation Router Processor Metrics

Overview This chapter describes the Fuse Mediation Router metrics collected for each processor in a route and supported control actions.

Processor Metrics The Processors service collects metrics related to each processor in a Camel route. For example, `bean1` identifies the following processor:

```
Jane-Murpheys-MacBook-Pro.local ServiceMix 4.4.0-fuse "bean1"  
Jane-Murpheys-MacBook-Pro.local/217-camel-3 Processors
```

Figure 3.1. `bean1` Processor



Supported metrics are:

- `Availability`.

This metric is of category *Availability* and collection type *Dynamic*.

Indicates the percentage of time the selected processor was available during a given time slice.

See [Figure 2.2 on page 16](#) for the key to the availability status icons

The following metrics are of category *Throughput* and collection type *Trends Up*:

- `ExchangesCompleted`

Indicates the total number of exchanges the selected processor has processed successfully.

By default, Fuse HQ displays the absolute number of completed exchanges since processor start-up or the last `reset` operation.

- `ExchangesFailed`

Indicates the total number of exchanges the selected processor has failed to process.

By default, Fuse HQ displays the absolute number of failed exchanges since processor start-up or the last `reset` operation.

- `ExchangesTotal`

Indicates the total number of exchanges, passed or failed, the selected processor has processed.

By default, Fuse HQ displays the absolute number of exchanges that have occurred since processor start-up or the last `reset` operation.

The following metrics are of category *Throughput* and collection type *Dynamic*.

Because Trends Up metrics continuously increase, the rate of change becomes more important to track. So Fuse HQ automatically calculates and creates a secondary *per Minute* metric for each Trends Up metric. By default, the *per Minute* Throughput metrics are enabled and charted.

- `ExchangesCompleted` per Minute

Indicates the number of exchanges the processor has successfully processed per minute, since route start-up or the last `reset` operation.

- `ExchangesFailed` per Minute

Indicates the number of exchanges that the processor has failed to process per minute, since route start-up or the last `reset` operation.

- `ExchangesTotal` per Minute

Indicates the number of exchanges, passed or failed, that the processor has processed per minute, since route start-up or the last `reset` operation.

The following metrics are of category *Performance* and collection type *Dynamic*:

- `MinProcessingTime`

Indicates the shortest time, in milliseconds, to process an exchange since processor start-up or the last `reset` operation.

- `MeanProcessingTime`

Indicates the average processing time, in milliseconds, for all exchanges processed since processor start-up or the last `reset` operation.

- `MaxProcessingTime`

Indicates the longest time, in milliseconds, to process an exchange since processor start-up or the last `reset` operation.

- `TotalProcessingTime`

Indicates the total processing time, in milliseconds, of all exchanges processed since processor start-up or the last `reset` operation.

- `LastProcessingTime`

Indicates the time, in milliseconds, it took the selected processor to process the last exchange.

Processor Control Actions

Processor control actions enable you to control individual processors in a route and to retrieve performance and statistics data from them.

Supported control actions are:

- `start`

The `start` operation starts up the selected processor immediately.

- `stop`

The `stop` operation stops the selected processor immediately.

This operation is useful when you need to load updates made to the processor's configuration.

- `reset`



Warning

The reset operation clears all counters, thus all statistics, for the selected processor.

- `getCamelID`

The `getCamelID` operation returns the ID of the `CamelContext` in which the selected processor is running; for example,

```
217-camel-3
```

- `getRouteId`

The `getRouteId` operation returns the ID of the route in which the selected processor is running; for example,

```
route1
```

- `getProcessorId`

The `getProcessorId` operation returns the selected processor's `ID`; for example,

```
bean1
```

