Version number 9.50

Data Model and Query API

Diagnostics

MICRO FOCUS

Document release date: May 2018 Software release date: May 2018

Legal Notices

Disclaimer

Certain versions of software and/or documents (“Material”) accessible here may contain branding from Hewlett-Packard Company (now HP Inc.) and Hewlett Packard Enterprise Company. As of September 1, 2017, the Material is now offered by Micro Focus, a separately owned and operated company. Any reference to the HP and Hewlett Packard Enterprise/HPE marks is historical in nature, and the HP and Hewlett Packard Enterprise/HPE marks are the property of their respective owners.

Warranty

The only warranties for Seattle SpinCo, Inc. and its subsidiaries (“Seattle”) products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. Seattle shall not be liable for technical or editorial errors or omissions contained herein. The information contained herein is subject to change without notice.

Restricted Rights Legend

Confidential computer software. Except as specifically indicated, valid license from Seattle required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Copyright Notice

Trademark Notices

Adobe™ is a trademark of Adobe Systems Incorporated.

Microsoft® and Windows® are U.S. registered trademarks of Microsoft Corporation.

UNIX® is a registered trademark of The Open Group.

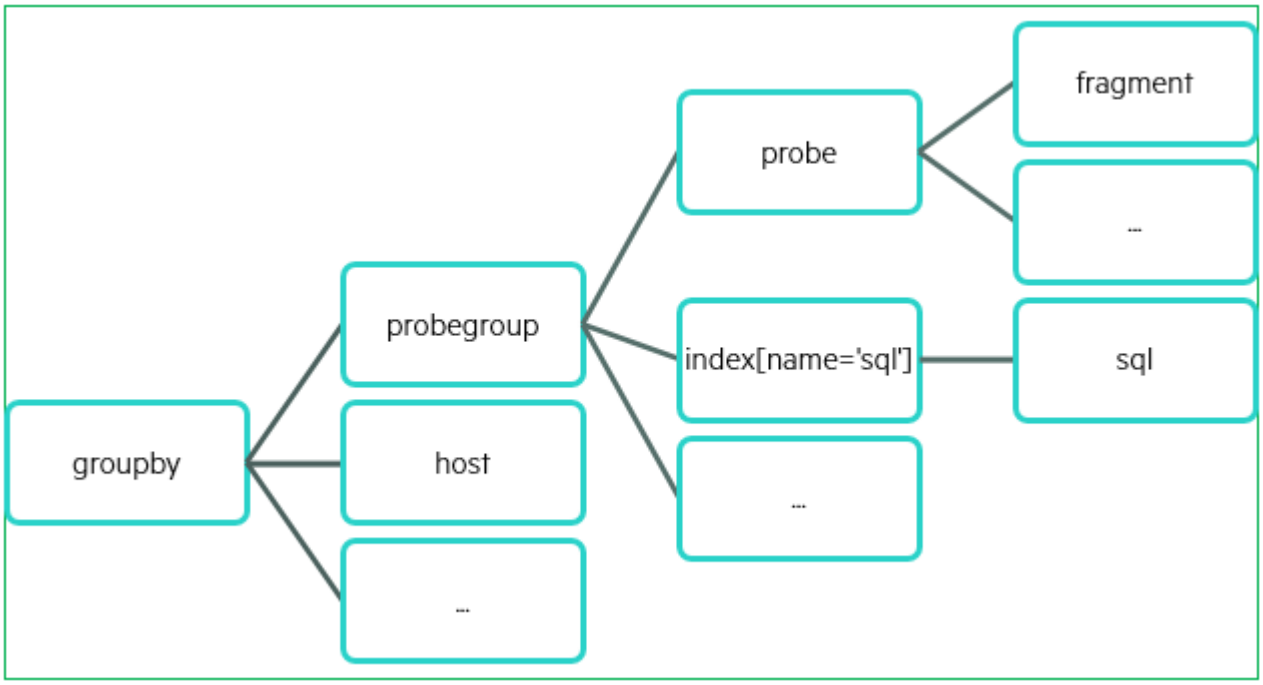
Oracle and Java are registered trademarks of Oracle and/or its affiliates.

Contents

The Diagnostics Data Model

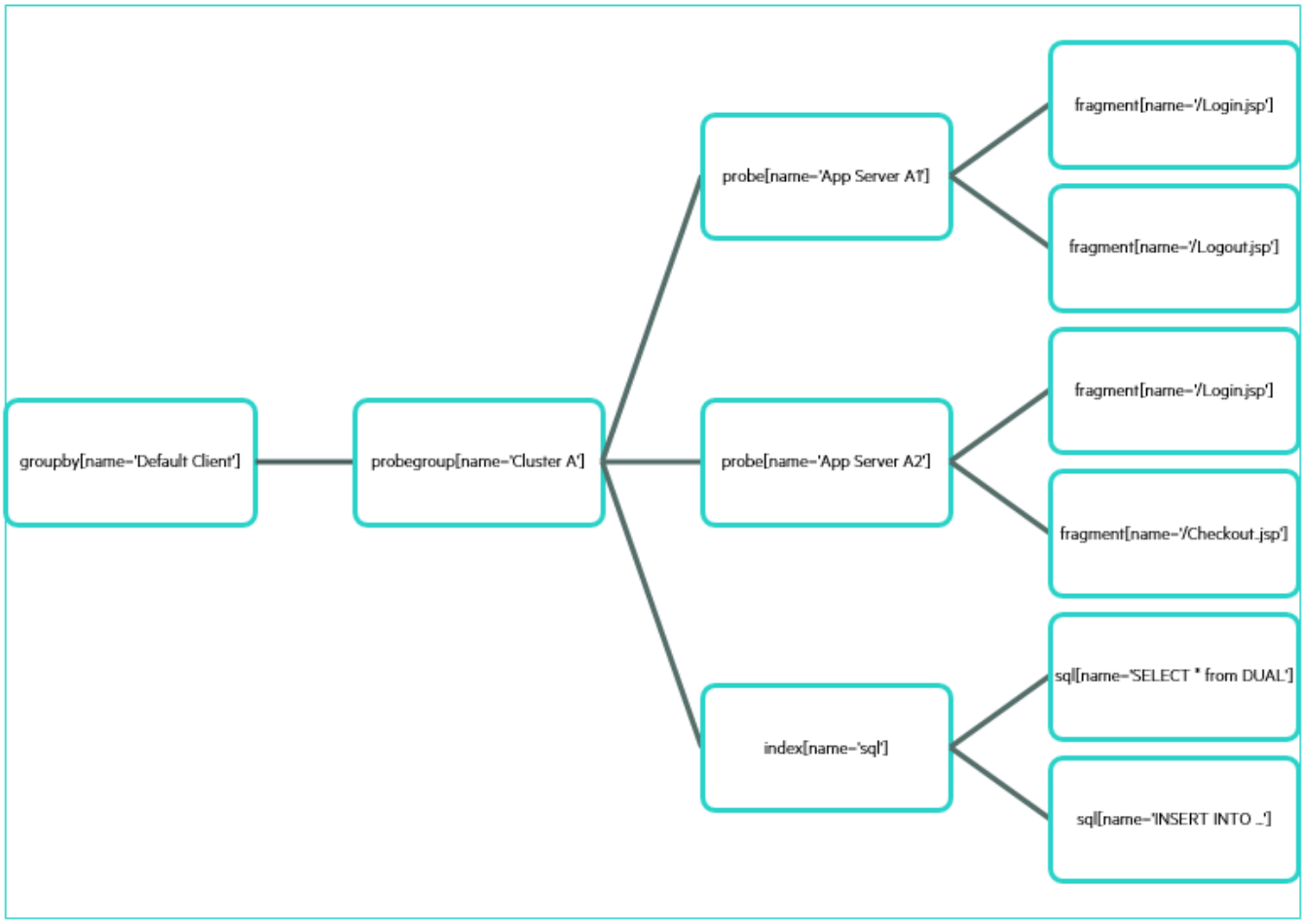
Diagnostics uses a hierarchical data model to store its entities like Server Requests, Probes, Hosts, etc., very similar to an XML document.

The diagram below shows the structure or schema of the data model:



The root of the tree is always the groupby entity which allows Diagnostics to support multi-tenancy. Special “index” nodes don’t have any metrics associated with them but act as a container to reduce the amount of nodes at the same level in the tree. For example, instead of putting all SQL statements (sql entity) directly under probegroup, the intermediary index entity groups all SQL statements under it, thus reducing the number of children under probegroup.

Here is an example of an instantiated model with actual data:



The default groupby is “Default Client”. Diagnostics uses the “Mercury System” groupby for internal Diagnostics Server monitoring. For LoadRunner and PC, temporary groupbys are created that contain the data for a run.

The tree structure allows for aggregated metrics. In the above example, the SQL index provides aggregate SQL statement metrics across all probes (App Server A1 and App Server A2). In other words, the metrics associated with SQL statement “SELECT * from DUAL” might have been executed on both application servers and the SQL statement latency is averaged across both servers.

A good way to explore the Diagnostics data model is via the Query page. It can be accessed by navigating to:
`http://<commander>:2006/query/?&response_format=html`

Attributes

Each entity has a set of attributes like name,probeType which uniquely identify an instance of an entity. For example, a groupby only has a name attribute which uniquely identifies it. Diagnostics uses brackets to specify a list of attributes.

For example:

```
groupby[name='Default Client']
groupby[name='Mercury System']
host[name='www.mycompany.com', systemgroup='Default']
probe[name='App Server A1', probeType='Java', hostname='www.mycompany.com',
systemgroup='Default']
```

The attributes are dependent on the entity type.

In addition to these per entity attributes which uniquely identify an instance, another set of general purpose attributes are available and associated with an entity instance. For example, the probe entity sometimes has an additional attribute that specifies the application server type (WebLogic, WebSphere …). These attributes have an info prefix in their name, like "info:0:app_server_name". The **Query** page shows all available attributes in the "Node Fields" section when selecting an entity instance.

Path

A 'path' in Diagnostics terminology allows querying and selecting instances. Since Diagnostics uses a hierarchical model, a path is formed by concatenating entity names and using attributes to do the filtering. This is similar to XPath expressions.

For example, to select the www.mycompany.com host instance, the path would look like this:

```
/groupby[name='Default Client']/host[name='www.mycompany.com',
systemgroup='Default']
```

This exactly identifies the instance for host www.mycompany.com.

Omitting attributes allows querying for multiple instances. For example, to get a list of all hosts, the path would look like this:

```
/groupby[name='Default Client']/host
```

To get a list of all Java probes, the path would look like this:

```
/groupby[name='Default Client']/probegroup/probe[probeType='Java']
```

In addition to the equals match (=), Diagnostics supports additional functions in a path expression:

Function	Description
equals	Equality function (like =)
notequals	Checks for nonequality
contains	Matches a partial string. For example: host[contains(name, 'mycompany')] returns all hosts that have 'mycompany' in their name
matches	Match based on a regular expression (slower than contains but more powerful)
startswith	Matches beginning of a string
endswith	Matches end of a string

All of the above functions can be used inside the brackets and are written in the form 'function(attribute, value)":

```
/groupby[name='Default Client']
  /probegroup[startswith(name,'Cluster')]
    /probe[contains(name,'Billing'),equals(probeType,'Java')]
```

(Note: For readability the path example above is split up and each entity is on its own line. When specifying a path in Diagnostics, it must not contain any CR and/or LFs).

Entity Types

This section lists and describes the most commonly used entity types. Since new types are added to the data model with each release it is best to use the **Query** page to browse the model.

groupby – Root node that partitions all instances

probegroup – Probe group (typically used to define a cluster of probes)

host – Host entities from the system metrics collector

probe – The probes or application servers

fragment – All Server Requests

txn – BPM transactions

index – Various indexes are used to group data. They typically don't have any metrics associated with them

Example Path Expressions

Example Path Expression	Description
/groupby[name='Default Client'] /probegroup /probe	All probes across all probe groups
/groupby[name='Default Client'] /probegroup /probe /fragment	All server requests across all probe groups and probes
/groupby[name='Default Client'] /probegroup /index[name='rollup_fragment'] /fragment	Server Requests that are rolled up across all probes of the same probe group
/groupby[name='Default Client'] /probegroup /probe /index[name='services'] /service	Web services across all probes
/groupby[name='Default Client'] /probegroup /probe[probeType='Oracle']	All Oracle probes (from collector)
/groupby[name='Default Client'] /probegroup /probe[probeType='SqlServer']	All SQL Server probes (from collector)
/groupby[name='Default Client'] /host	All hosts
/groupby[name='Default Client'] /txn	All BPM transactions
/groupby[name='Default Client'] /index[name='apps'] /app /app_metrics	Application metrics

Diagnostics Query API

The query API can be used to retrieve data from Diagnostics in XML and CSV format. The API is a standard HTTP based API that takes its arguments as query parameters and returns data in the HTTP reply body. The query API is protected by basic HTTP authentication. Any Diagnostics user with view permissions can access the query API.

Diagnostics summarized data is kept at certain granularities like 5m, 20m, 1h etc. (more information about granularities can be found in the Diagnostics Server Installation and Administration Guide). This summarized data is displayed in the tables in the Enterprise UI. In addition to summarized data, Diagnostics keeps trend data. Trends are displayed as the charts in the Enterprise UI.

For example the following URL retrieves the last 5m of summarized data for the probe 'Foo' in CSV format:

```
http://<commander>:2006/query/?action=summary&granularity=[name='5m']&response_format=
excel&path=/groupby[name='Default Client']/probegroup[equals(name,'Default')]/probe[equals(name,'Foo')]
```

Note the separator for the CSV format is a tab character.

The following table shows the possible parameters and values that can be passed to the /query URL.

Parameter Name	Values	Comment
action	Summary trend.	Action specifies what type of data should be returned. Either summarized data or trend data.
granularity=[name='<value>']	5m, 20m, 1h, 6h, 1d, 7d, 1M, 3M, 1Y	Granularity of the returned data. Last 5m, last 20m etc.
granularity=[name='<value>',start='<start timestamp>',end='<end timestamp>']	5m, 20m, 1h, 6h, 1d, 7d, 1M, 3M, 1Y start timestamp and end timestamp	Returns data for the specified time range and granularity. The timestamp is in milliseconds since epoch (as defined by Java's System.currentTimeMillis()). Note, due to internal bucketing, sometimes more data is returned than specified by the time range.
response_format	excel xml	Either excel (CSV format) or XML
server	Diagnostics server id	This parameter is required when querying for trend data. It is not required for summary data.
path	Entity path	Either a fully qualified path to an entity or a partial path. See the Path section in this document for more information or use the Query page to create a path.

The following example shows how to query for trend data. In contrast to the summary data, the trend data query requires that a specific metric is specified for which the data should be retrieved. For example:

```
http://<commander>:2006/query/?action=trend&granularity=[name='5m']&response_format
=
excel&path=/groupby[name='Default Client']/probegroup[equals(name,'Default')]/probe
[equals(name,'Foo')]/metric[equals(name,'HeapFree'),type='average']
```

Note: The date time in the CSV format of the returned trend data is in Excel format. To convert it to a timestamp in milliseconds use the following formula:

```
d = d - 25569;
d = d * 24 * 60 * 60;
d = d * 1000;
timeInMilliseconds = Math.round(d)
```

When working with the query API it is recommended to try out the queries first via the Query Page.

Threshold and Alert Configuration API

Diagnostics provides the ability to script the setting of thresholds and alerts on entities. This functionality is provided from the Enterprise UI via Maintenance -> thresholding and via a REST API.

There are three statements possible in a script to:

- Set or remove threshold on an entity
- Set an alert on an entity
- Remove an alert on an entity

Tip: The log/server.log file contains an audit trail of successful threshold or alert rule operations.

API for Setting Thresholds

The basic form of a threshold statement is

```
threshold metric:”<valid metric name>", [type:"<type of metric>",]
(critical:"value", | warning:"value",) path:"<query format path to the entity>"

threshold default:"", metric:”<valid metric name>”, [type:”<type of metric>”,]
path:”<query format path to the entity>”

threshold none: "", metric:”<valid metric name>”, [type:”<type of metric>”,]
path:”<query format path to the entity>”
```

A threshold setting script statement can consist of multiple statements separated by a carriage return.

Metric – Name of a metric. Available metric names can be found in etc/metrics.config on a probe system.

type – Metric type is not required, by default "average" is assumed. You can select any one of the following comma-separated values: average, average_cpu, count, cpu_count, exception_count, exclusive_average, exclusive_cpu, exclusive_total, exclusive_total_cpu, latency_contribution, load, max, min, percent_contribution, rel_std_dev, std_dev, sum_of_squares, throughput, timeout_count, total, total_cpu, value_over_threshold.

critical|warning – Threshold value. You can enter a value for critical or warning or both. The value contains a number and the units ("1s"). Units can be: ms, s, m, h, d, w, y, B, KB, MB, GB, /sec, /min, /hr, /day. Note that before support for both critical and warning thresholds, there would have been only one value for the threshold.

path – The path to the entity using the Query API format. See previous sections to find the path to the entity instance within the Diagnostics data model.

Example - to set a critical threshold of 10 and a warning threshold of 5 on metric “GC Time Spent in Collections” on probe “App Server A1”:

threshold metric:"GC Time Spent in Collections", critical:”10”, warning:”5”, path:"/groupby[name='Default Client']/probegroup/probe[name='App Server A1']"

Example - to set a critical latency 1 sec threshold on a server request:

threshold metric:"latency", critical:”1s”,

path:"/groupby[name='Default Client']/probegroup/probe[name='App Server A1']/fragment[info:0:displayName='StockQuotes::GetQuote000']"

default – Use the threshold default syntax to set a threshold to the default threshold value.

Example - threshold default:"", metric:"latency", path:"/groupby[name='Default Client']/probegroup/probe[name='App Server A1']/fragment[info:0:displayName='StockQuotes::GetQuote000']"

none – Use the threshold none syntax to remove/delete thresholds.

Example - threshold none: "", metric:"GC Time Spent in Collections", path:"/groupby[name='Default Client']/probegroup/probe[name='App Server A1']"

A negative number (-10) is interpreted as a threshold below the specified value.

API for Setting an Alert Rule

Like thresholds, the alerts operate on an entity path and have the following parameters:

```
alert name:"<any name>", description:"<any description>", metric:"<valid metric name>", [type:"<type of metric>",] (snmp:<true|false>, | smtp:"<email address>", | scripts:"<script name>",) [on_red:<true|false>,] [on_yellow:<true|false>,] [on_green:<true|false>,] [on_no_data:<true|false>,]  path:"<query format path to the entity>"
```

```
alert delete:””, [metric:”<valid metric name>",] [type:”< type of metric>”,] path:"<entity path>"
```

name – Name for the alert rule.

description – Description of the alert rule.

metric – Name of a metric. Available metric names can be found in etc/metrics.config on a probe system.

type – (Optional) Metric type is not required, by default "average" is assumed. You can select any one of the following comma-separated values: average, average_cpu, count, cpu_count, exception_count, exclusive_average, exclusive_cpu, exclusive_total, exclusive_total_cpu, latency_contribution, load, max, min, percent_contribution, rel_std_dev, std_dev, sum_of_squares, throughput, timeout_count, total, total_cpu, value_over_threshold.

Configure at least one of the following actions to be executed on an alert triggering:

snmp – (Optional) Indicates if you want an snmp notification sent on alert triggering. If you want to set the snmp option in the script statement you must first have configured the snmp communication (server, ports...) in the Alert Properties page.

smtp – (Optional) The email address for sending an email notification on alert triggering. If you want to set use the smtp option in the script statement you must first have configured the smtp communication (server, ports...) in the Alert Properties page.

scripts – (Optional) The scripts parameter indicates the scripts to execute on alert triggering. The user-defined scripts must be located in the <Diagnostics_server_install_dir>/scripts/alerts directory. If you want to use the scripts option you must first enable script execution on alerts in the Alerts Properties page.

The following four entries are optional and indicate when you want an alert notification to be triggered. Note that on_red and on_green are set to true by default so alerts will trigger when status is red and when status returns to green

on_red – By default set to true. Indicates that you want an alert notification when status is red.

on_yellow – Indicates that you want an alert notification when status is yellow.

on_green – By default set to true. Indicates that you want an alert notification when status returns to green.

on_no_data – Indicates that you want an alert to trigger on no data received.

The path uses the query format to specify an entity instance:

path – The path to the entity using the Query API format. See previous sections for information on how to find the path to the entity instance within the Diagnostics data model.

Example - to set an alert with SMTP (mail) and SNMP (trap) notification on “GC Time Spent in Collections” on probe “App Server A1” to fire on red (critical) and green (normal):

```
alert name:"GC alert", description:"Alert on GC", smtp:"foo@bar.com", snmp:true,
metric:"GC Time Spent in Collections",
path:"/groupby[name='Default+Client']/probegroup/probe[name='App Server A1']"
```

delete – Use the alert delete syntax to remove an alert.

Example:

```
alert delete:"", metric:"GC Time Spent in Collections",
path:"/groupby[name='Default+Client']/probegroup/probe[name='App Server A1']"
```

Using Script Statements to Set Thresholds and Alerts

Diagnostics provides a utility to set thresholds and alerts using script statements.

You can access the page for creating script statements from the URL **http://<Diagnostics Server>:2006/thresholding/script.jsp** or from the Diagnostics Server Administration page where you select **Configure Diagnostics > thresholding**. You can use this page to set thresholds, set alerts, delete thresholds, delete alerts and export currently defined thresholds and alerts.

REST API

The REST API takes a script and executes it repeatedly until at least one script statement returns a success.

The URL for the REST API is:

http://<commander>:2006/rest/thresholding/once/customer/{customer}/script/{script}

{customer} is typically “Default Client” and {script} is a unique script name.

The script executes every 15s until one of the statements is executed successfully. This allows the setting of threshold and alert rules for entities that haven’t yet reported in to Diagnostics (e.g., newly deployed probes).

In case it is important to know whether all statements of the script have been executed, you can create one single script per statement.

If a script hasn’t been successfully executed within 7d (7 days) it will be deleted.

Send Us Feedback:

Let us know how we can improve your experience with the Diagnostics Data Model and Query API Guide.
Send your email to: docteam@microfocus.com