HP Diagnostics

for the Windows©, Unix and Linux operating systems

Software Version: 9.12

Data Model and Query API

Document Release Date: November 2011

Software Release Date: November 2011

Legal Notices

Warranty

Restricted Rights Legend

Copyright Notices

Trademark Notices

Acknowledgements

This product includes software developed by the Apache Software Foundation (http://www.apache.org). This product includes software developed by the JDOM Project (http://www.jdom.org). This product includes software developed by the MX4J project (http://mx4j.sourceforge.net).

Support

Visit the HP Software Support web site at:

www.hp.com/go/hpsoftwaresupport

This Web site provides contact information and details about the products, services, and support that HP Software offers.

HP Software online support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the support web site to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Download software patches
- Manage support contracts
- Look up HP support contacts
- Review information about available services
- Enter into discussions with other software customers
- Research and register for software training

To find more information about access levels, go to:

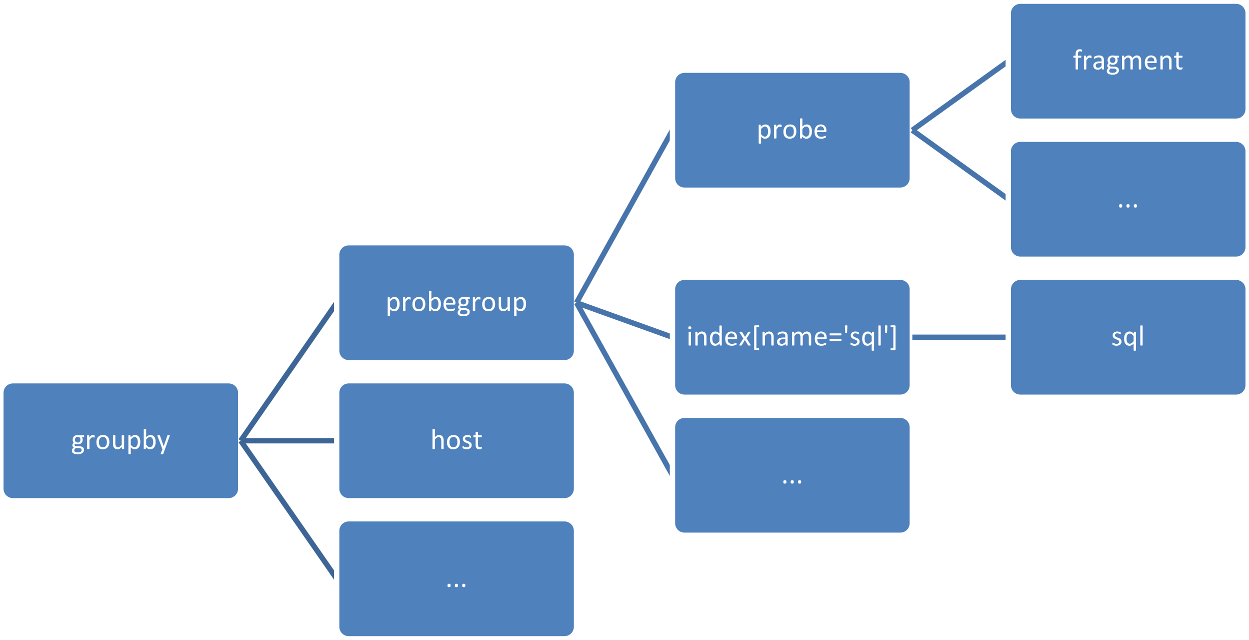
http://h20230.www2.hp.com/new_access_levels.jsp

Contents

The HP Diagnostics Data Model

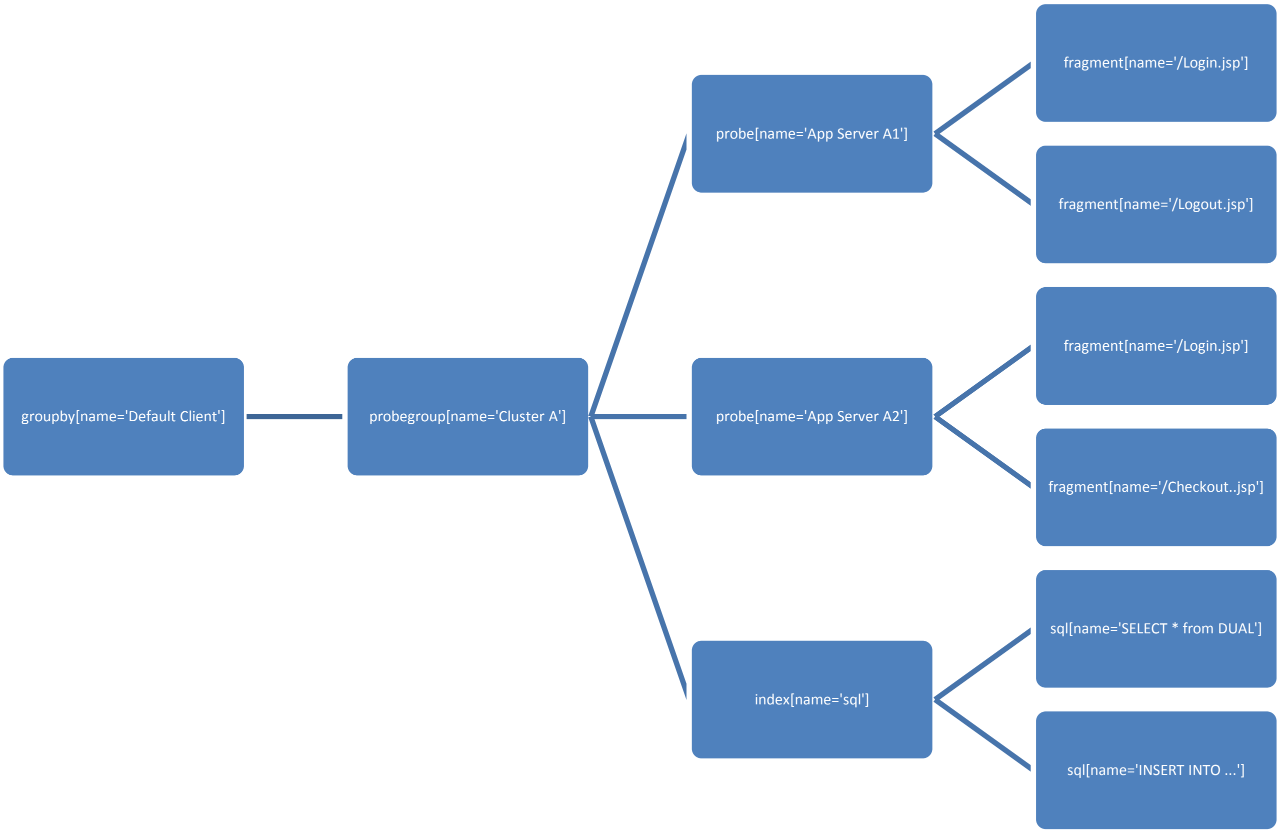
Diagnostics uses a hierarchical data model to store its entities like Server Requests, Probes, Hosts, etc., very similar to an XML document.

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:


groupby[name='Default Client']
probegroup[name='Cluster A']
probe[name='App Server A1']
probe[name='App Server A2']
index[name='sql']
fragment[name='/Login.jsp']
fragment[name='/Logout.jsp']
fragment[name='/Login.jsp']
fragment[name='/Checkout..jsp']
sql[name='SELECT * from DUAL']
sql[name='INSERT INTO ...']

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.hp.com', systemgroup='Default']

probe[name='App Server A1', probeType='Java',
hostname='www.hp.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.hp.com host instance, the path would look like this:

```
/groupby[name=’Default Client’]/host[name=’www.hp.com’,
systemgroup=’Default’]
```

This exactly identifies the instance for host www.hp.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, ‘hp’)] returns all hosts that have ‘hp’ 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

The two major Linux distributors, Red Hat and Novell/SUSE LINUX, have adopted an "Enterprise Linux" approach in an effort to provide stable releases and reduce the frequency of new releases. With this enterprise approach, the goal is to place new releases on an 18- to 24-month schedule rather than a 4- to 9-month schedule. As Linux continues to mature, however, incremental updates will still be necessary. Both Red Hat and Novell/SUSE LINUX provide quarterly updates that contain bug fixes, new features, additional security, and new hardware support.

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

Example Path Expression	**Description**
/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 *HP Diagnostics Installation and Configuration 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.

Parameter Name	Values	Comment
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,'Fo
o')]/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.