

# OPTIMIZE

**MERCURY BUSINESS AVAILABILITY CENTER™**  
Integrating with Third-Party Applications

**MERCURY™**  
BUSINESS TECHNOLOGY OPTIMIZATION



# **Mercury Business Availability Center**

Integrating with Third-Party Applications

Version 6.5

Document Release Date: October 15, 2006

---

**MERCURY™**

## Mercury Business Availability Center, Version 6.5 Integrating with Third-Party Applications

This manual, and the accompanying software and other documentation, is protected by U.S. and international copyright laws, and may be used only in accordance with the accompanying license agreement. Features of the software, and of other products and services of Mercury Interactive Corporation, may be covered by one or more of the following patents: United States: 5,511,185; 5,657,438; 5,701,139; 5,870,559; 5,958,008; 5,974,572; 6,137,782; 6,138,157; 6,144,962; 6,205,122; 6,237,006; 6,341,310; 6,360,332; 6,449,739; 6,470,383; 6,477,483; 6,549,944; 6,560,564; 6,564,342; 6,587,969; 6,631,408; 6,631,411; 6,633,912; 6,694,288; 6,738,813; 6,738,933; 6,754,701; 6,792,460 and 6,810,494. Australia: 763468 and 762554. Other patents pending. All rights reserved.

U.S. GOVERNMENT RESTRICTED RIGHTS. This Software Documentation is a “commercial item” as defined at 48 C.F.R. 2.101 (October 1995). In accordance with 48 C.F.R. 12.212 (October 1995), 48 C.F.R. 27.401 through 27.404 and 52.227-14 (June 1987, as amended) and 48 C.F.R. 227.7201 through 227.7204 (June 1995), and any similar provisions in the supplements to Title 48 of the C.F.R. (the “Federal Acquisition Regulation”) of other entities of the U.S. Government, as applicable, all U.S. Government users acquire and may use this Documentation only in accordance with the restricted rights set forth in the license agreement applicable to the Computer Software to which this Documentation relates.

Mercury, Mercury Interactive, the Mercury logo, the Mercury Interactive logo, LoadRunner, WinRunner, SiteScope and TestDirector are trademarks of Mercury Interactive Corporation and may be registered in certain jurisdictions. The absence of a trademark from this list does not constitute a waiver of Mercury's intellectual property rights concerning that trademark.

All other company, brand and product names may be trademarks or registered trademarks of their respective holders. Mercury disclaims any responsibility for specifying which marks are owned by which companies or which organizations.

Mercury provides links to external third-party Web sites to help you find supplemental information. Site content and availability may change without notice. Mercury makes no representations or warranties whatsoever as to site content or availability.

Mercury Interactive Corporation  
379 North Whisman Road  
Mountain View, CA 94043  
Tel: (650) 603-5200  
Toll Free: (800) TEST-911  
Customer Support: (877) TEST-HLP  
Fax: (650) 603-5300

© 2005-2006 Mercury Interactive Corporation, All rights reserved

If you have any comments or suggestions regarding this document, please send them by e-mail to [documentation@mercury.com](mailto:documentation@mercury.com).

---

# Table of Contents

Welcome to Integrating with Third-Party Applications .....	v
How This Guide is Organized.....	v
Who Should Read This Guide .....	vi
Getting More Information .....	vi

## **PART I: INTRODUCTION**

Chapter 1: Introduction to Integrating with Third-Party Applications .....	3
---	---

## **PART II: WORKING WITH DATA APIS**

Chapter 2: Working with the Generic Data Engine API .....	7
Using the Generic Data Engine API .....	8
Data Returned .....	10
Chapter 3: The OpenAPI Query Builder.....	13
Introducing the OpenAPI Query Builder .....	13
Creating Queries Using the OpenAPI Query Builder .....	14
Chapter 4: Creating Sample Queries .....	19
Queries for the Generic Data Engine .....	19
Generic Data Engine Query Examples .....	24
Legacy Queries.....	25
Chapter 5: Working with the CMDB API .....	39
Using the CMDB API.....	40
Calling the Web Service .....	41
CMDB Parameter Format .....	42
CMDB Module Methods.....	45
Use Cases and Examples.....	50

**PART III: WORKING WITH MERCURY BUSINESS AVAILABILITY CENTER  
EMS ADAPTERS**

<b>Chapter 6: Introducing Mercury Business Availability Center EMS Adapters.....</b>	<b>73</b>
<b>Chapter 7: Sending Mercury Business Availability Center Alerts to BMC PATROL.....</b>	<b>75</b>
About Sending Mercury Business Availability Center Alerts to BMC PATROL.....	75
Installing the Mercury Business Availability Center SNMP Adapter on the BMC PATROL Agent(s) .....	76
Installation Troubleshooting .....	76
<b>Chapter 8: Sending Mercury Business Availability Center Alerts to CA Unicenter .....</b>	<b>79</b>
About Sending Mercury Business Availability Center Alerts to CA Unicenter.....	79
Sending a Mercury Business Availability Center Alert as an SNMP Trap .....	80
Sending a Mercury Business Availability Center Alert Using the Unicenter cawto Command.....	81
Configuration Troubleshooting .....	83
<b>Chapter 9: Sending Mercury Business Availability Center Alerts to HP Openview VantagePoint Operations .....</b>	<b>87</b>
About Sending Mercury Business Availability Center Alerts to HP Openview VantagePoint Operations.....	88
Verifying the Presence of the HP OVO Agent.....	88
Installing the HP OVO Agent on the Core Server Machine .....	89
Assigning and Distributing the opcmsg Template to the Mercury Business Availability Center Host.....	90
Sending a Mercury Business Availability Center Alert Using the HP OVO Agent opcmsg Command.....	91
<b>Chapter 10: Sending Mercury Business Availability Center Alerts to Tivoli TEC.....</b>	<b>93</b>
About Sending Mercury Business Availability Center Alerts to Tivoli TEC.....	93
Setting Up Tivoli TEC.....	94
Sending a Mercury Business Availability Center Alert Using the Tivoli End Point postemsg Command.....	95
<b>Index.....</b>	<b>97</b>

---

# Welcome to Integrating with Third-Party Applications

This guide introduces third-party integrations, describes how to work with the Generic Data Engine API to extract data from Mercury Business Availability Center for use with third-party or custom tools, and explains how to integrate alerts generated by Mercury Business Availability Center into an EMS console.

## How This Guide is Organized

The guide contains the following parts:

### **Part I Introduction**

Introduces the various types of third-party integrations that can be performed with Mercury Business Availability Center.

### **Part II Working with Data APIs**

Describes how to use the Mercury Business Availability Center generic data engine API to extract data from Mercury Business Availability Center for use with third-party or custom tools as well as how to use the CMDB API to read/write data from/to the CMDB.

### **Part III Working with Mercury Business Availability Center EMS Adapters**

Describes how to integrate Mercury Business Availability Center with different EMS applications, to send Mercury Business Availability Center alerts to the EMS console.

## Who Should Read This Guide

This guide is intended for the following users of Mercury Business Availability Center:

- ▶ Mercury Business Availability Center administrators
- ▶ Mercury Business Availability Center platform administrators
- ▶ Mercury Business Availability Center application administrators
- ▶ Mercury Business Availability Center data collector administrators

Readers of this guide should be knowledgeable about enterprise system administration and Mercury Business Availability Center.

## Getting More Information

For information on using and updating the Mercury Business Availability Center Documentation Library, reference information on additional documentation resources, typographical conventions used in the Documentation Library, and quick reference information on deploying, administering, and using Mercury Business Availability Center, refer to *Getting Started with Mercury Business Availability Center*.



# Part I

---

## Introduction



# 1

---

## Introduction to Integrating with Third-Party Applications

Mercury Business Availability Center enables you to perform several types of integrations. The list below describes each integration and where to get more information.

- ▶ **Generic Data Engine API.** Enables extraction of data from the Mercury Business Availability Center database for use with third-party or custom tools. For details, see “Working with Data APIs” on page 5.
- ▶ **CMDB API.** Enables writing configuration item definitions and topological relations to the CMDB (Configuration Management Database), and querying the information with TQL and ad hoc queries. For details, see “Working with the CMDB API” on page 39.
- ▶ **EMS adapters.** Enable integrating alerts generated by Mercury Business Availability Center into an EMS console. For details, see “Working with Mercury Business Availability Center EMS Adapters” on page 71.
- ▶ **EMS integrations using SiteScope Integration Monitors.** Enable collecting data from EMS systems using SiteScope Integration Monitors and feeding the data into Mercury Business Availability Center. For details, see “Working with SiteScope Integration Monitors” in *Configuring SiteScope Monitors*.
- ▶ **Deep transaction tracing.** Provides a monitoring layer for collecting information about the behavior of Business Process Monitor transactions within the target machine, achieved through integration with Bristol's TransactionVision application. The resulting information is displayed in the Deep Transaction Tracing view in Dashboard. For details, see “Administering Deep Transaction Tracing” in *Application Administration*.



# Part II

---

## Working with Data APIs



# 2

---

## Working with the Generic Data Engine API

---

**Note to Mercury Managed Services customers:** For details on how to use the Generic Data Engine API in a Managed Services environment, contact Mercury Managed Services Support.

---

This chapter explains how to work with the Generic Data Engine API to extract data from Mercury Business Availability Center for use with third-party or custom tools.

<b>This chapter describes:</b>	<b>On page:</b>
Using the Generic Data Engine API	8
Data Returned	10

## Using the Generic Data Engine API

---

**Note:** Users of the Generic Data Engine API should be familiar with:

- SQL
  - the SOAP specification (if using the Web Service)
  - an object-oriented programming language such as C++ or Java (if using the Web Service)
  - Mercury Business Availability Center administration and applications
- 

The Generic Data Engine enables you to retrieve data from Mercury Business Availability Center profile databases by sending a query using the following methods:

- **Web browser.** The request is sent as an HTML query and the data is returned as HTML or as a CSV (Comma Separated Values) file that can be opened with Microsoft Excel or processed with a custom tool.
- **Web Service.** The return object contains the data in CSV format.

### Creating Queries

You can create queries to send to the Generic Data Engine using the OpenAPI Query Builder or by building them manually.

The OpenAPI Query Builder provides a graphical interface that facilitates the building of HTML queries, which can then be sent to the Generic Data Engine using a Web browser. For details, see Chapter 3, “The OpenAPI Query Builder.”

For information on creating queries manually and query syntax and limitations, see Chapter 4, “Creating Sample Queries.”



## Getting Metadata on the Samples

To create a query, you must know the data representation of the sample. For information on commonly queried samples, see “Samples” in *Reference Information*.

Users with special reporting needs can retrieve a list of all samples and their fields using the MBean Inspector. Access the MBean Inspector page by entering the following URL in your browser:

```
http://<server>[:port]/jmx-console/HtmlAdaptor?action=inspectMBean&name=Topaz%3AService%3DMeta-Data+Manager
```

The default port number is 8080. If this port is incorrect, consult your system administrator for the correct port number.

On the MBean Inspector page, click the **Invoke** button next to the operation **showMetaDataDBMapping**. The bean returns a list of fields in each sample.

## Permissions

Either the System Viewer or Superuser role is required to access the data. If the user name passed in the query does not have these rights, the response is an authentication error.

## Calling the Web Service

The Generic Data Engine Web Service enables submitting a query consisting of a username, password, and an SQL-like select statement. The engine returns an error description if it cannot parse the statement or if there is a problem running the query. If there is no error, the results of the query are returned.

SOAP programmers can access the WSDL at:

```
http://<server>[:port]/topaz/gdeopenapi/services/GdeWsOpenAPI?wsdl
```

The port specification is only necessary for non-standard installations. Consult your system administrator for the correct port number.

For information about creating the query, see Chapter 4, “Creating Sample Queries.”

## Querying with a Browser

When querying with a browser, the `getData` service is called with the URL:

```
http://<server>[:port]/topaz/gdeopenapi/GdeOpenApi?method=getData&user=<username>&password=<password>&query=<query>
```

or with the optional result type parameter:

```
http://<server>[:port]/topaz/gdeopenapi/GdeOpenApi?method=getData&user=<username>&password=<password>&query=<query>&resultType=csv
```

The port specification is only necessary for non-standard installations. Consult your system administrator for the correct port number.

The default return type is HTML. If `resultType=csv` is specified, a comma separated values file is returned.

For information about creating the query, see “Creating Sample Queries” on page 19.

## Data Returned

The same data is returned whether the request is made from a browser or with the Web Service. For the Web Service, the data resides in the return object, and with a browser, the data resides in the response body.

### Web Service Return Object

The Web Service return object contains the following:

- **retval.** The data or an error message.
- **errorCode.** The error code (type int). Possible error codes are:
  - 0 - Success
  - 100 - Authorization error

- 101 - Processing error
- 102 - Open API has been disabled
- **origRowCount**. The actual number of rows the query should have returned (type int). If the number of rows to be returned exceeds the maximum, the **origRowCount** field is set to the actual number of rows that the query would have returned had the maximum not been exceeded.

## Web Browser Response Body

When the query is submitted from a browser, the response CSV or HTML contains the data, or error code and message. If the number of rows to be returned exceeds the maximum, the last row of the data is **Returned X of Y rows**, where *X* is the number of rows returned and *Y* is the actual number of rows that fulfil the conditions of the query. If there is an error at the engine level, the HTTP success code is returned, but the body of the response is **<error code>**, **<error message>**.

## Setting Maximum Number of Rows Returned

The maximum number of rows returned can be configured in the Infrastructure Settings Manager. To modify the maximum number of rows returned, select **Admin > Platform > Setup and Maintenance > Infrastructure Settings**, choose **Foundations**, select **Generic Data Engine Open API**, and locate the **Maximum Rows** entry in the Generic Data Engine Open API Settings table. Modify the value to the required number of rows.



# 3

---

## The OpenAPI Query Builder

---

**Note to Mercury Managed Services customers:** This section is not relevant to Mercury Managed Services customers.

---

This chapter describes how to use the OpenAPI Query Builder to create Generic Data Engine API queries.

This chapter describes:	On page:
Introducing the OpenAPI Query Builder	13
Creating Queries Using the OpenAPI Query Builder	14

### Introducing the OpenAPI Query Builder

The OpenAPI Query Builder is a Web interface that assists in creating queries, which can then be sent via a Web browser to the Generic Data Engine to extract data from the Mercury Business Availability Center profile database.

#### Accessing the OpenAPI Query Builder

The OpenAPI Query Builder is located in the User Reports tab of the following Mercury Business Availability Center applications:

- Service Level Management
- End User Management

► System Availability Management

To access the OpenAPI Query Builder, in any of the above applications, select the User Reports tab and click **OpenAPI Query Builder**.

### Permissions

To access the OpenAPI Query Builder page, a user must have Administrator or Superuser permissions.

For a query to access the data using the Generic Data Engine API, the user and password parameters passed in the query must be those of a user with either System Viewer or Superuser permissions. (For details on setting permissions in the Permissions Manager, see “Configuring User Permissions” in *Platform Administration*.) When creating a query using the OpenAPI Query Builder, the user and password parameters of the current user are automatically added to the query URL. Thus if the current user is not a System Viewer or Superuser, the login parameters in the query must be modified to those of either a System Viewer or Superuser. Otherwise, the response to the query is an authentication error.

## Creating Queries Using the OpenAPI Query Builder

The following steps describe how to use the OpenAPI Query Builder to create a query that can be sent via a Web browser to the Generic Data Engine API. For a list of query limitations, see “Query Limitations” on page 21.

- 1 Select the time range and granularity of the data to be extracted from the database.

Using the time range selector bar, specify the time range for which you want to extract data.

If you want returned data grouped according to a specific granularity setting (for example, if you want data for the past day grouped per hour), specify a granularity setting and select **Return data per specified granularity**.

For details on using the time range selector bar, see “Choosing the Tracking Range and Granularity” in *Working with Applications*.

**2** Select the type of data you want to extract from the database.

From the **Sample type** list, select a data sample type. For a description of the available sample types, see “Samples” in *Reference Information*.

**3** Select the data fields of the chosen sample type that you want to extract from the database and the functions to perform on the data. For detailed descriptions of the available data fields per sample type, see “Samples” in *Reference Information*.

---

**Tip:** When adding fields to the query, select them in the order in which you want the data columns to appear, from left to right, in the returned output.

---

**a** From the **Field name** list, select a data field.

**b** Where applicable, from the **Function** list select the function that you want applied to the data. For example, to return average response time, select **Response Time** from the **Field name** list and **AVG** from the **Function** list.

For a description of the supported functions, see “Supported SQL Syntax” on page 20.

**c** If required, modify the alias name for the field in the **Alias name** box to suit the function applied to the field. For example, if you chose **AVG** from the **Function** list, modify the alias name to say Avg Response Time.

**d** Click **Add** to add the data field to the query.

**e** Optionally, once a data field is added to the query, you can edit the field alias and field formula values directly in the **Fields to Return** table. For example, to display returned response time in seconds rather than milliseconds, edit the field formula for Response Time, **dResponseTime**, as follows: `dResponseTime / 1000`.

**f** Repeat the above steps for each data field you want to add to the query.



**g** If required, click the **Delete** button to delete a data field from the query.

#### 4 Apply filters to the extracted data.

---

**Tip:** Click **Preview** to see all returned results before applying filters. To limit the number of returned values, select a short time range.

---

---

**Note:**

- ▶ When manually defining a filter that consists of strings containing white space or special characters (for example, `where bb_guid IN (a b, c)`), you must enclose the white space or special character string with quotes (for example, `where bb_guid IN ('a b', c)`). When you create filters on the Filter Builder page, Mercury Business Availability Center automatically adds the quotes. Special characters are defined as any characters other than digits, letters, and the following characters: "\_", "\$", "#".
  - ▶ When defining a filter that consists of strings containing one or more single quote characters, you must add a second single quote character beside each instance. For example, change `szTransactionName = ('Login_to_O'brien')` to `szTransactionName = ('Login_to_O''brien')`.
- 

- a** Click **Filter Builder** to apply filters to the data that the query will extract.
  - b** From the **Field** list, select a field to which you want to apply a filter. For example, select **Profile Name** to filter the query to only return data for specific profiles (rather than all profiles in the database), or select **Response Time** to filter the query to return only response time data above or below a specific value (rather than all response times).
  - c** In the **Operator** list, select the required operator. For a description of the supported operators, see “Supported SQL Syntax” on page 20.
- 

**Note:** The LIKE operator does not work for the Profile Name field.

---



- d** In the **Value** box, specify the required value. For example, if you are filtering the query by transaction name, specify a specific transaction. If you use the LIKE operator, you can use the \* wildcard character to return like results (for example, a value of \*westcoast\* would return all transactions whose name included the string **westcoast**).

When available, use the **Add Values** button to select from a predefined list of values. The Add Values button is enabled for fields whose lookup value is defined as **true** in the sample metadata in the database (generally for final, discreet values).

- e** Click **And** to add additional **And** filters. Click **Add 'Or' Expression** to add additional **Or** filters.
- f** Click **OK** when all filters are defined.
- g** If required, click **Clear All** on the main OpenAPI Query Builder page to clear the filter string.

#### 5 Select additional display options.

- From the **Return results in time zone** list, select the time zone in which you want the data displayed.
- From the **Result format** list, select whether you want data returned in HTML format or CSV format.

#### 6 Verify the query.

- Click **Preview** for a preview of the data that the query will return.
- Click **Display Query** to view the query URL.
- Click **Clear Form** to clear the query and start again.

#### 7 Run the query.

- a** Click **Display Query** to view the query URL.
- b** Copy and paste the query URL into the Web browser address box and run it.

Depending on the selected result format, the data will be returned either in the Web browser window or in a CSV file (which you can save as required).

---

**Tip:** Save the query URLs you build in a separate file, since the queries built using the OpenAPI Query Builder cannot be saved.

---

# 4

---

## Creating Sample Queries

This chapter explains how to manually create queries used by the Generic Data Engine API to extract data from Mercury Business Availability Center profile databases.

This chapter describes:	On page:
Queries for the Generic Data Engine	19
Generic Data Engine Query Examples	24
Legacy Queries	25

### Queries for the Generic Data Engine

The Generic Data Engine is a Mercury Business Availability Center component that creates an interface for samples so that they appear as virtual database tables and the sample fields appear as virtual database table fields. This component allows you to query samples using SQL. These queries can be used with the Web Service and to create custom Excel reports.

For more information, see:

- Chapter 2, “Working with the Generic Data Engine API”
- Chapter 3, “The OpenAPI Query Builder”
- “Samples” in *Reference Information*
- “Integrating Microsoft Excel Reports in Mercury Business Availability Center” in *Working with Applications*

## Supported SQL Syntax

The language supported is a subset of SQL and supports these keywords, modifiers, and operators:

- SELECT
- WHERE
- FROM
- TOP
- HAVING
- Aliasing with the AS keyword
- Logical operators OR , AND , NOT
- DISTINCT modifier
- IN operator. Inner selects can be used to return the values for the IN operator.
- BETWEEN operator
- IS NULL (IS NOT NULL is not supported)
- LIKE. Note that the wildcard character is the asterisk (\*). Do not use the percent sign (%). The asterisk can not be used by itself (LIKE \*). It must be used with other characters.
- Comparators: =, IS, !=, <>, >, >=, <, <=
- Functions: MAX, MIN, SUM, COUNT, AVG, STDDEV, SUMOFSQR, LOG, CEIL, FLOOR, MOD, SQRT, REPLACENULL, IF, and the “byTime Function” described on page 23.
- ORDER BY and the ASC and DESC modifiers

## Query Limitations

The following limitations apply to queries submitted to the service:

- ▶ Composite expressions in the SELECT clause using parenthesis are not supported. For example, you cannot select (a+b)/c. Use of parentheses for function arguments is supported, for example, max(a+b).
- ▶ Only one monitor type can be selected in a single query.
- ▶ The asterisk (\*) is not supported as a wildcard character except in combination with the LIKE operator. It is supported as the multiplication operator.
- ▶ Inner selects and joins are not supported, with one exception: an inner select can be used to return the values for an IN clause.
- ▶ The ORDER BY clause requires a column number, for example ORDER BY 1. ORDER BY column name is not supported.
- ▶ The engine requires that queries contain a time limitation in the WHERE clause.
- ▶ The GROUP BY clause is not supported. It is unnecessary because the engine treats all fields that do not have an aggregate function as GROUP BY fields.
- ▶ When manually defining a filter that consists of strings containing white space or special characters (for example, where bb\_guid IN (a b, c)), you must enclose the white space or special character string with quotes (for example, where bb\_guid IN ('a b', c)). When you create filters on the Filter Builder page, Mercury Business Availability Center automatically adds the quotes. Special characters are defined as any characters other than digits, letters, and the following characters: "\_", "\$", "#".
- ▶ When defining a filter that consists of strings containing one or more single quote characters, you must add a second single quote character beside each instance. For example, change szTransactionName = ('Login\_to\_O'brien') to szTransactionName = ('Login\_to\_O''brien').
- ▶ The columns in the returned data are labeled Column 0, Column 1, and so on. To return meaningful column labels, use the SQL AS operator. For example, Select time\_stamp as TimeStamp. With this use of the AS operator, the column label is TimeStamp.

## Date-Time Values

Time in queries and return data is specified in seconds since January 1, 1970. You can use Microsoft Excel to understand the time values.

Time is most commonly used for time stamp fields.

To get a GMT time for use in a query, enter the date and time in a Date-formatted cell and in another cell, formatted as General, enter the formula:

$=(<date\ cell> - 25569) * 86400$

To correct for a local time zone, add the time zone offset times 3600 seconds to the result. For example, for Central Europe (GMT + 2):

$=(<date\ cell> - 25569) * 86400 + (2 * 3600)$

To view a time value from a query as a GMT date in Excel, use a Date format for the cell and enter the formula:

$=<time\ stamp> / 86400 + 25569$

To correct for a local time zone, subtract the time zone offset times 3600 seconds from the time stamp. For example, for the Eastern United States, standard time (GMT - 3):

$=(<time\ stamp> - (-3 * 3600)) / 86400 + 25569$

## byTime Function

The Generic Data Engine SQL supports the function **byTime**, which returns data grouped by time periods. For example, if you build a query that returns the average response time of a transaction for the past day, without the **byTime** function one value would be returned; using the **byTime** function, you could request to view the average response time of the transaction for each hour of the past day, in which case a value would be returned for each hour of the past 24 hours.

The function syntax is:

**byTime**(*<timefield >*, *<step value>*, *<number of step>*, *<offset>*)

Argument	Description
<i>timefield</i>	Usually a timestamp field
<i>step value</i>	One of: -1 - Time not set 10 - Second 20 - Minute 30 - Hour 40 - Day 50 - Week 60 - Month 70 - Quarter 80 - Year
<i>number of step</i>	The number of the units specified in <i>step value</i> to group.
<i>offset</i>	Time zone offset from GMT in hours. Positive numbers indicate time zones East of GMT. Negative numbers indicate time zones West of GMT.

For example, to return one value for each 3 days, corrected to two hours East of GMT:

**byTime**(time\_stamp, 40, 3, 2)

## Configuration

You can configure the Generic Data Engine Open API options at **Admin > Platform > Setup and Maintenance > Infrastructure Settings > Foundations > Generic Data Engine Open API**. You can set the maximum number of data rows returned, as well as disable use of the Generic Data Engine Open API.

## Generic Data Engine Query Examples

Below are several examples of query URLs that retrieve different types of data from the database.

### Example of ss\_t Sample

This example illustrates retrieving the average value for SiteScope samples on a given measurement and monitor.

```
http://myServer/topaz/gdeopenapi/GdeOpenApi?method=getData&user=admin  
&password=admin&query=select szMeasurementName, szMonitorName,  
avg(dValue) from ss_t where u_iStatus=1 and time_stamp > 123456 and  
szMeasurementName = 'myMeasurementName' and szMonitorName =  
'myMonitorName'
```

### Example of trans\_t Sample

This example illustrates retrieving the average response time, grouped by minutes and offset to GMT + 3 for Springfield\_infra\_ems\_login transactions in the Springfield\_Location profile on for a given period from BPM data.

```
http://myServer/topaz/gdeopenapi/GdeOpenApi?method=getData&user=admin  
&password=admin&query=select byTime(time_stamp, 20, 1, 3.0), profile_name  
as ProfileName, szTransactionName as TransactionName,  
AVG(dResponseTime) from trans_t where time_stamp>=1126594800.64 and  
time_stamp<1126596000.64 and profile_name='Springfield_Location' and  
szTransactionName='Springfield_infra_ems_login'
```



### Example of rum\_server\_t Sample

This example illustrates retrieving a list of all rum\_server\_t samples in a given day that failed on a specified server:

```
http://myServer/topaz/gdeopenapi/GdeOpenApi?method=getData&user=admin
&password=admin&query=select time_stamp, engine_name, server_name from
rum_server_t where availability=0 and total_hits > 0 and time_stamp >
1041379200 and time_stamp < 1136197020 and server_name =
'myServerName'
```

### Example of rum\_page\_t Sample

This example illustrates retrieving the total server time for each URL as measured by RUM.

```
http://myServer/topaz/gdeopenapi/GdeOpenApi?method=getData&user=admin
&password=admin&query=select page_url, sum(tot_server_time) from
rum_page_t where time_stamp > 1041379200 and time_stamp < 1136197020
&resultType=csv
```

## Legacy Queries

From Mercury Business Availability Center version 6.0, it is possible to write SQL queries directly on samples, which serve as virtual database tables, as described in “Queries for the Generic Data Engine” on page 19. To maintain older reports, use the information in this section.

### Structure of the Query

The query begins with the URL of the OpenAPI.jsp:  
 http://<server\_name>/topaz/openapi/OpenAPI.jsp

Following this is a series of query **parameter=value** pairs. The pairs are separated with an ampersand (&). You build the query using required and optional parameters.

You can also add filter parameters. For details, see “Filter Parameters” on page 36.

► **Required Parameters.** The following parameters are required:

<i>username</i>	The name of the Mercury Business Availability Center user running the query.
<i>password</i>	The password of the Mercury Business Availability Center user running the query
<i>function</i>	The type of the data to receive. One of: <b>transactions</b> – Business Process Monitor response time data <b>transactions_breakdown</b> – Business Process Monitor transaction breakdown data <b>sitescope</b> – SiteScope measurement data <b>routings</b> – WebTrace routing data <b>hops</b> – WebTrace hop data <b>transactions_error</b> – Business Process Monitor transaction error data <b>alerts</b> – Business Process Monitor alert data <b>components_breakdown</b> – Business Process Monitor component breakdown data
<i>rowDataTypes</i>	The type of aggregation to use. One of: <b>row</b> – raw data <b>hour</b> – hourly aggregated data <b>day</b> – daily aggregated data The type of aggregation cannot be a larger unit than the one specified with the optional parameter, stepValueType.

<i>profileIds</i>	<p>The IDs of the profiles for which to get data. Separate multiple IDs with a comma (,).</p> <p>For example: <code>profileIds=2,7,9</code></p> <p>To extract a profile's ID from the database:</p> <ol style="list-style-type: none"> <li>1 Open the page <code>http://&lt;servername&gt;:8080/jmx-console/index.jsp</code> in a Web browser, where <b>&lt;servername&gt;</b> represents the name of the Mercury Business Availability Center server.</li> <li>2 In the <b>Topaz</b> list, click the link <b>service=OAPI Services</b>.</li> <li>3 Invoke <b>com.mercury.infra.db.tools.oapiservices.ProfileMapping getProfiles()</b> to retrieve the profile ID information.</li> </ol>
<p>Time range</p> <p>either <i>from</i> and <i>to</i> or <i>lastPeriod</i></p>	<p>Either use <i>from</i> and <i>to</i> to specify the start and end of the period, or use <i>lastPeriod</i> to specify the period or periods ending at the time the query is called.</p> <p>The time format for <i>from</i> and <i>to</i> is: <code>dd/MM/yyyy HH:mm:ss</code></p> <p><i>lastPeriod</i> is one of:</p> <p><b>hour</b> <b>day</b> <b>week</b> <b>month</b> <b>quarter</b> <b>year</b></p> <p>To specify more than one unit of <i>lastPeriod</i>, use the parameter <code>numStepUnit</code> together with <i>lastPeriod</i>.</p>

► **Optional Parameters.** The following parameters are optional:

<i>numStepUnit</i>	Used with <i>lastPeriod</i> to indicate the number of periods included in the data. For example, get data for the last two days with: <code>lastPeriod=day&amp;numStepUnit=2</code>
<i>resultType</i>	The output file format. One of: <b>html</b> <b>xml</b> <b>csv</b> The default is <b>html</b> .
<i>stepValueType</i>	The unit of time by which the data is grouped. One of: <b>minute</b> <b>hour</b> <b>day</b> <b>week</b> <b>month</b> Use <i>stepValueType</i> in conjunction with <i>stepValue</i> to see a larger grouping than that specified with <i>rowDataTypes</i> .
<i>stepValue</i>	The number of the time units specified by <i>stepValueType</i> by which to group. For example <code>stepValueType=hour&amp;stepValue=8</code> groups the data by 8-hour periods. The default is 1.
<i>dateFormat</i>	The format for date value output (that is, the way the date and time are displayed in the report). For example, <code>MM/dd/yyyy%20HH:mm:ss</code> The “%20” indicates a space. (A space is ASCII character number 32, written 20 in hexadecimal.)
<i>timeZone</i>	A time zone specification in the form: <code>&amp;timezone=&lt;time zone value from below time zone list&gt;</code> For example: <code>&amp;timezone=Europe/London</code>

**Time Zone List**

ACT	Antarctica/McMurdo	Etc/GMT0
AET	Antarctica/Palmer	Etc/Greenwich
AGT	Antarctica/Rothera	Etc/UCT
ART	Antarctica/South_Pole	Etc/UTC
AST	Antarctica/Syowa	Etc/Universal
Africa/Abidjan	Antarctica/Vostok	Etc/Zulu
Africa/Accra	Arctic/Longyearbyen	Europe/Amsterdam
Africa/Addis_Ababa	Asia/Aden	Europe/Andorra
Africa/Algiers	Asia/Almaty	Europe/Athens
Africa/Asmera	Asia/Amman	Europe/Belfast
Africa/Bamako	Asia/Anadyr	Europe/Belgrade
Africa/Bangui	Asia/Aqtau	Europe/Berlin
Africa/Banjul	Asia/Aqtobe	Europe/Bratislava
Africa/Bissau	Asia/Ashgabat	Europe/Brussels
Africa/Blantyre	Asia/Ashkhabad	Europe/Bucharest
Africa/Brazzaville	Asia/Baghdad	Europe/Budapest
Africa/Bujumbura	Asia/Bahrain	Europe/Chisinau
Africa/Cairo	Asia/Baku	Europe/Copenhagen
Africa/Casablanca	Asia/Bangkok	Europe/Dublin
Africa/Ceuta	Asia/Beirut	Europe/Gibraltar
Africa/Conakry	Asia/Bishkek	Europe/Helsinki
Africa/Dakar	Asia/Brunei	Europe/Istanbul
Africa/Dar_es_Salaam	Asia/Calcutta	Europe/Kaliningrad
Africa/Djibouti	Asia/Choibalsan	Europe/Kiev

Africa/Douala	Asia/Chongqing	Europe/Lisbon
Africa/El_Aaiun	Asia/Chungking	Europe/Ljubljana
Africa/Freetown	Asia/Colombo	Europe/London
Africa/Gaborone	Asia/Dacca	Europe/Luxembourg
Africa/Harare	Asia/Damascus	Europe/Madrid
Africa/Johannesburg	Asia/Dhaka	Europe/Malta
Africa/Kampala	Asia/Dili	Europe/Minsk
Africa/Khartoum	Asia/Dubai	Europe/Monaco
Africa/Kigali	Asia/Dushanbe	Europe/Moscow
Africa/Kinshasa	Asia/Gaza	Europe/Nicosia
Africa/Lagos	Asia/Harbin	Europe/Oslo
Africa/Libreville	Asia/Hong_Kong	Europe/Paris
Africa/Lome	Asia/Hovd	Europe/Prague
Africa/Luanda	Asia/Irkutsk	Europe/Riga
Africa/Lubumbashi	Asia/Istanbul	Europe/Rome
Africa/Lusaka	Asia/Jakarta	Europe/Samara
Africa/Malabo	Asia/Jayapura	Europe/San_Marino
Africa/Maputo	Asia/Jerusalem	Europe/Sarajevo
Africa/Maseru	Asia/Kabul	Europe/Simferopol
Africa/Mbabane	Asia/Kamchatka	Europe/Skopje
Africa/Mogadishu	Asia/Karachi	Europe/Sofia
Africa/Monrovia	Asia/Kashgar	Europe/Stockholm
Africa/Nairobi	Asia/Katmandu	Europe/Tallinn
Africa/Ndjamena	Asia/Krasnoyarsk	Europe/Tirane
Africa/Niamey	Asia/Kuala_Lumpur	Europe/Tiraspol
Africa/Nouakchott	Asia/Kuching	Europe/Uzhgorod

Africa/Ouagadougou	Asia/Kuwait	Europe/Vaduz
Africa/Porto-Novo	Asia/Macao	Europe/Vatican
Africa/Sao_Tome	Asia/Macau	Europe/Vienna
Africa/Timbuktu	Asia/Magadan	Europe/Vilnius
Africa/Tripoli	Asia/Makassar	Europe/Warsaw
Africa/Tunis	Asia/Manila	Europe/Zagreb
Africa/Windhoek	Asia/Muscat	Europe/Zaporozhye
America/Adak	Asia/Nicosia	Europe/Zurich
America/Anchorage	Asia/Novosibirsk	GB
America/Anguilla	Asia/Omsk	GB-Eire
America/Antigua	Asia/Oral	GMT
America/Araguaina	Asia/Phnom_Penh	GMT0
America/Aruba	Asia/Pontianak	Greenwich
America/Asuncion	Asia/Pyongyang	HST
America/Atka	Asia/Qatar	Hongkong
America/Barbados	Asia/Qyzylorda	IET
America/Belem	Asia/Rangoon	IST
America/Belize	Asia/Riyadh	Iceland
America/Boa_Vista	Asia/Riyadh87	Indian/Antananarivo
America/Bogota	Asia/Riyadh88	Indian/Chagos
America/Boise	Asia/Riyadh89	Indian/Christmas
America/Buenos_Aires	Asia/Saigon	Indian/Cocos
America/Cambridge_Bay	Asia/Sakhalin	Indian/Comoro
America/Cancun	Asia/Samarkand	Indian/Kerguelen
America/Caracas	Asia/Seoul	Indian/Mahe
America/Catamarca	Asia/Shanghai	Indian/Maldives

America/Cayenne	Asia/Singapore	Indian/Mauritius
America/Cayman	Asia/Taipei	Indian/Mayotte
America/Chicago	Asia/Tashkent	Indian/Reunion
America/Chihuahua	Asia/Tbilisi	Iran
America/Cordoba	Asia/Tehran	Israel
America/Costa_Rica	Asia/Tel_Aviv	JST
America/Cuiaba	Asia/Thimbu	Jamaica
America/Curacao	Asia/Thimphu	Japan
America/Danmarkshavn	Asia/Tokyo	Kwajalein
America/Dawson	Asia/Ujung_Pandang	Libya
America/Dawson_Creek	Asia/Ulaanbaatar	MET
America/Denver	Asia/Ulan_Bator	MIT
America/Detroit	Asia/Urumqi	MST
America/Dominica	Asia/Vientiane	MST7MDT
America/Edmonton	Asia/Vladivostok	Mexico/BajaNorte
America/Eirunepe	Asia/Yakutsk	Mexico/BajaSur
America/El_Salvador	Asia/Yekaterinburg	Mexico/General
America/Ensenada	Asia/Yerevan	Mideast/Riyadh87
America/Fort_Wayne	Atlantic/Azores	Mideast/Riyadh88
America/Fortaleza	Atlantic/Bermuda	Mideast/Riyadh89
America/Glace_Bay	Atlantic/Canary	NET
America/Godthab	Atlantic/Cape_Verde	NST
America/Goose_Bay	Atlantic/Faeroe	NZ
America/Grand_Turk	Atlantic/Jan_Mayen	NZ-CHAT
America/Grenada	Atlantic/Madeira	Navajo
America/Guadeloupe	Atlantic/Reykjavik	PLT



America/Guatemala	Atlantic/South_Georgia	PNT
America/Guayaquil	Atlantic/St_Helena	PRC
America/Guyana	Atlantic/Stanley	PRT
America/Halifax	Australia/ACT	PST
America/Havana	Australia/Adelaide	PST8PDT
America/Hermosillo	Australia/Brisbane	Pacific/Apia
America/Indiana/Indiana polis	Australia/Broken_Hill	Pacific/Auckland
America/Indiana/Knox	Australia/Canberra	Pacific/Chatham
America/Indiana/Mareng o	Australia/Darwin	Pacific/Easter
America/Indiana/Vevay	Australia/Hobart	Pacific/Efate
America/Indianapolis	Australia/LHI	Pacific/Enderbury
America/Inuvik	Australia/Lindeman	Pacific/Fakaofu
America/Iqaluit	Australia/Lord_Howe	Pacific/Fiji
America/Jamaica	Australia/Melbourne	Pacific/Funafuti
America/Jujuy	Australia/NSW	Pacific/Galapagos
America/Juneau	Australia/North	Pacific/Gambier
America/Kentucky/Louis ville	Australia/Perth	Pacific/Guadalcanal
America/Kentucky/Mont icello	Australia/Queensland	Pacific/Guam
America/Knox_IN	Australia/South	Pacific/Honolulu
America/La_Paz	Australia/Sydney	Pacific/Johnston
America/Lima	Australia/Tasmania	Pacific/Kiritimati
America/Los_Angeles	Australia/Victoria	Pacific/Kosrae
America/Louisville	Australia/West	Pacific/Kwajalein
America/Maceio	Australia/Yancowinna	Pacific/Majuro

America/Managua	BET	Pacific/Marquesas
America/Manaus	BST	Pacific/Midway
America/Martinique	Brazil/Acre	Pacific/Nauru
America/Mazatlan	Brazil/DeNoronha	Pacific/Niue
America/Mendoza	Brazil/East	Pacific/Norfolk
America/Menominee	Brazil/West	Pacific/Noumea
America/Merida	CAT	Pacific/Pago_Pago
America/Mexico_City	CET	Pacific/Palau
America/Miquelon	CNT	Pacific/Pitcairn
America/Monterrey	CST	Pacific/Ponape
America/Montevideo	CST6CDT	Pacific/Port_Moresby
America/Montreal	CTT	Pacific/Rarotonga
America/Montserrat	Canada/Atlantic	Pacific/Saipan
America/Nassau	Canada/Central	Pacific/Samoa
America/New_York	Canada/East-Saskatchewan	Pacific/Tahiti
America/Nipigon	Canada/Eastern	Pacific/Tarawa
America/Nome	Canada/Mountain	Pacific/Tongatapu
America/Noronha	Canada/Newfoundland	Pacific/Truk
America/North_Dakota/Center	Canada/Pacific	Pacific/Wake
America/Panama	Canada/Saskatchewan	Pacific/Wallis
America/Pangnirtung	Canada/Yukon	Pacific/Yap
America/Paramaribo	Chile/Continental	Poland
America/Phoenix	Chile/EasterIsland	Portugal
America/Port-au-Prince	Cuba	ROK
America/Port_of_Spain	EAT	SST

America/Porto_Acre	ECT	Singapore
America/Porto_Velho	EET	SystemV/AST4
America/Puerto_Rico	EST	SystemV/AST4ADT
America/Rainy_River	EST5EDT	SystemV/CST6
America/Rankin_Inlet	Egypt	SystemV/CST6CDT
America/Recife	Eire	SystemV/EST5
America/Regina	Etc/GMT	SystemV/EST5EDT
America/Rio_Branco	Etc/GMT+0	SystemV/HST10
America/Rosario	Etc/GMT+1	SystemV/MST7
America/Santiago	Etc/GMT+10	SystemV/MST7MDT
America/Santo_Domingo	Etc/GMT+11	SystemV/PST8
America/Sao_Paulo	Etc/GMT+12	SystemV/PST8PDT
America/Scoresbysund	Etc/GMT+2	SystemV/YST9
America/Shiprock	Etc/GMT+3	SystemV/YST9YDT
America/St_Johns	Etc/GMT+4	Turkey
America/St_Kitts	Etc/GMT+5	UCT
America/St_Lucia	Etc/GMT+6	US/Alaska
America/St_Thomas	Etc/GMT+7	US/Aleutian
America/St_Vincent	Etc/GMT+8	US/Arizona
America/Swift_Current	Etc/GMT+9	US/Central
America/Tegucigalpa	Etc/GMT-0	US/East-Indiana
America/Thule	Etc/GMT-1	US/Eastern
America/Thunder_Bay	Etc/GMT-10	US/Hawaii
America/Tijuana	Etc/GMT-11	US/Indiana-Starke
America/Tortola	Etc/GMT-12	US/Michigan
America/Vancouver	Etc/GMT-13	US/Mountain

America/Virgin	Etc/GMT-14	US/Pacific
America/Whitehorse	Etc/GMT-2	US/Pacific-New
America/Winnipeg	Etc/GMT-3	US/Samoa
America/Yakutat	Etc/GMT-4	UTC
America/Yellowknife	Etc/GMT-5	Universal
Antarctica/Casey	Etc/GMT-6	VST
Antarctica/Davis	Etc/GMT-7	W-SU
Antarctica/DumontDUrville	Etc/GMT-8	WET
Antarctica/Mawson	Etc/GMT-9	Zulu

## Filter Parameters

There can be one **filters** clause in the query. It is not required. If there is more than one filter in the clause, the filters are separated by a semicolon (;).

A filter consists of a report column name, a filter type, the number of patterns in the list, and a list of patterns. Each element in a filter is separated by a semicolon (;).

For example:

```
filters=MonitorTitle;in;2;9,12;Measurement;in;1;18
```

means:

```
select where (MonitorTitle equals 9 or 12) and (Measurement equals 18)
```

The following filter parameters are available:

General Filter Types	<p>These types can be used with numerical or text columns.</p> <p><b>in</b> – Data is included in the report if it equals one of the items in the list of patterns.</p> <p><b>not_in</b> – Data is included in the report if it does not equal any item in the list of patterns.</p>
Text Filter Types	<p>This type can be used with text columns.</p> <p><b>like</b> – Data is included in the report if it matches the pattern. When <b>like</b> is used, the number of patterns in the list must be one (1).</p> <p>The wildcard, asterisk (*), can be used with <b>like</b>.</p> <p>For example: filters=Host Name;like;1;ServerNum*</p>
Numeric Filter Types	<p>These types can be used with numerical columns.</p> <p><b>bigger</b></p> <p><b>smaller</b></p> <p><b>equals</b></p>

## Query Examples

Example 1:

```
http://<server_name>/topaz/openapi/OpenAPI.jsp?
username=fitzwilliam&password=darcy&function=transactions&profileIds=33&
lastPeriod=hour&rowData Type=raw&dateFormat=MM/dd/yyyy HH:mm:ss
```

Example 2:

```
http://<server_name>/topaz/openapi/OpenAPI.jsp?
username=fitzwilliam&password=darcy&function=transactions&profileIds=33&
from=10/12/2002 14:00:00&to=10/12/2002 15:00:00& rowData Type=raw&
dateFormat=MM/dd/yyyy%20HH:mm:ss
```

Example 3:

```
http://<server_name>/topaz/openapi/OpenAPI.jsp?username=fitzwilliam&password=darcy&function=sitescope&profileIds=35&lastPeriod=day&numStepUnit=2&&rowData Type=hour&dateFormat=MM/dd/yyyy%20HH:mm:ss&filters=MonitorTitle;in;1;9;Measurement;in;1;18&stepValueType=hour&stepValue=6
```

# 5

---

## Working with the CMDB API

---

**Note to Mercury Managed Services customers:** For details on how to use the CMDB API in a Managed Services environment, contact Mercury Managed Services Support.

---

This chapter explains how to work with the CMDB API to extract configuration data from Mercury Universal CMDB (Configuration Management Database) for use with third-party or custom tools, or to write data to the database.

<b>This chapter describes:</b>	<b>On page:</b>
Using the CMDB API	40
Calling the Web Service	41
CMDB Parameter Format	42
CMDB Module Methods	45
Use Cases and Examples	50

## Using the CMDB API

The CMDB API is a SOAP API used to integrate between the Mercury Universal CMDB and other applications. The API provides methods to add, remove, and update CIs (configuration items) and relations in the CMDB, and methods to query the CMDB base on TQL (topology query language).

The API also provides methods to query the CMDB. There are three types of queries:

- ▶ Queries that require topological information use a TQL, either previously defined in the CMDB, or passed by the client for ad hoc processing. Ad hoc queries are usually TQL definitions from the database that the client extracts and then uses the definition in the SOAP call.
- ▶ Queries on a single CI type based on a condition are used when topology is not important.
- ▶ The third type is queries on the topology surrounding a specific CI. This can be done with a TQL, but the APIs are faster and simpler. For example, you can query for all the neighbors or all descendents of a given CI that meet some criteria.

---

### Note:

The CMDB API is not compatible with an Apache AXIS client for versions later than AXIS 1.2.

Users of the CMDB API should be familiar with:

- ▶ the SOAP specification
  - ▶ an object-oriented programming language such as C++ or Java
  - ▶ Mercury Application Mapping
  - ▶ CMDB
  - ▶ Mercury Business Availability Center administration and applications
-



## Uses of the API

The API can be used to fulfill a number of business requirements. For example:

- ▶ A third party asset management tool may update the Mercury CMDB with information available only to that tool, thereby unifying the data with data collected by Mercury applications.
- ▶ A number of third party systems might populate the Mercury CMDB to create a central CMDB that can track changes and perform impact analysis.
- ▶ A third party system might create CIs and relations according to its business logic, then write the data to the CMDB to take advantage of the CMDB query capabilities.

## Permissions

To use the web service, the client must pass a user and password defined in the files in folder <install directory>\AppServer\webapps\cmdb\_openapi.war\WEB-INF\classes. Enter the data in files cmdb-open-api-users.properties and cmdb-open-api-roles.properties.

## Calling the Web Service

The CMDB Web Service enables calling server-side methods using standard SOAP programming techniques. The engine returns an error description if it cannot parse the statement, or if there is a problem invoking the method. If there is no error, the results of the invocation are returned.

SOAP programmers can access the WSDL at:

`http://<server>[:port]/cmdb_openapi/services/cmdb?wsdl`

The port specification is only necessary for non-standard installations. Consult your system administrator for the correct port number.

The URL for calling the service is:

`http://<server>[:port]/cmdb_openapi/services/cmdb`

## CMDB Parameter Format

All methods take a `cmdb-context` input parameter.

Data in the CMDB consists of data objects and links.

Data can be organized in `cmdb-graphs` and `tql-result-maps` as output and `data-containers` as input. The calling program should also define the `data-layout`.

TQL calculations return a `tql-result-map`.

CMDB API methods can throw a `SoapFault` exception. The error message, error code, and exception message fields may be populated. No specific CMDB exceptions are thrown.

All methods use CMDB IDs. An ID can be a real ID, a temporary ID, or an empty ID. A real ID is a string of 32 characters representing a hexadecimal number. A temporary ID can be any string. An empty ID means no value is assigned to the parameter.

A real ID identifies an entity that is stored in the CMDB.

A temporary ID does not represent an existing entity in the CMDB and must not be the same as any real ID. Note that when a temporary ID is used, if an existing data object can be identified by the other properties, that object is used as appropriate for the context as though it had been identified with a real ID.

Note that ID properties must be set during object or link creation and they are part of object or link ID calculation. Properties that have the `REQUIRED` attribute in the class module definition in the CIT manager must be set during object or link creation but are not part of object or link ID calculation.

### **cmdb-context**

CMDB API services get a `cmdb-context` parameter. This parameter specifies the caller and the customer to receive the request. The inner parameters are:

- `CUSTOMERID`. An integer that denotes the customer number for whom the request is invoked.

- **CALLERAPPLICATION.** A string that identifies the client that invokes the method. The caller application is used for logging and trouble-shooting. No server logic is performed with this parameter.

## **object**

An object is composed of an ID, an object type, and a collection of object properties.

The ID can be a real CMDB ID, a temporary ID, or an empty ID.

If a real CMDB ID is used, set the `ISCMDBID` flag to true.

The client can assign a temporary ID to an object that will be referenced by either of the two ends of a link. In this case, set the `ISCMDBID` flag to false.

An empty ID is used to enter a data object that is not referenced by any link in the CMDB. Where the object is not referenced, only the object type and properties are set.

## **link**

A link contains an ID, a link type, two end IDs, and a properties collection.

The link ID can be a real CMDB ID, a temporary ID, or empty. The link type is the name of the CMDB class from which the link is instantiated.

The two link end IDs may not be empty IDs. If the first is a real CMDB ID, set `ISEND1IDCMDBID` true. If the second is a real ID, set `ISEND2IDCMDBID` to true.

If you set the link properties only so that the link can be identified, and not to update data, the ID properties alone are sufficient. To add or update a link, the ID and those properties that have the `REQUIRED` attribute in the class module definition in the CIT manager must be entered. Other properties are optional. If a real CMDB ID is set in the link and it is used in a remove operation, properties are not used.

## **cmdb-graph**

A CMDB graph is a collection of objects and the links connecting these objects.

## **data-container**

A data container has `CMDBOBJECTS`, `CMDBLINKS`, and a `REFERENCEDOBJECTS` collection. The objects and links are written to the CMDB.

Referenced objects are used to define the ends of links or ID reference attributes of objects, for example, the `root_container`. The data in referenced objects is not written to the CMDB. If the client has no real IDs for the links' ends, it can set temporary IDs for the ends and define referenced objects for these temporary IDs.

## **data-layout**

The data layout defines the subset of objects and links and their properties to return. The data layout consists of `ELEMENT-TYPE-LAYOUT` entries for the objects and links.

An `ELEMENT-TYPE-LAYOUT` contains a `TYPE-TO-LAYOUT` map, the `ALLLAYOUT` flag, the `PROPQUALIFIERS` property qualifiers collection, and the `KEYS` key property names list.

The `TYPE-TO-LAYOUT` MAP is a list of `TYPE` (class name) and `ELEMENT-SIMPLE-LAYOUT` pairs. Each type can be linked to a specific simple layout.

If the `ALLLAYOUT` flag is true, all properties of all classes are returned, regardless of other layout settings. There is a significant performance penalty for setting this flag true. If the flag is false, all properties specified or implied by any of the other properties are returned.

The `PROPQUALIFIERS` property qualifiers list indirectly specifies the properties to be returned. Properties that have one of the specified qualifiers are returned.

The properties named in the `KEYS` list are returned regardless of their class name.

## **properties**

The collection of properties, called `PROPS` in the wsdl, is divided into collections of properties according to data type. For example: `StrProps` is a collection of strings, `IntProps` is a collection of integers.

## **tql-result-map**

The TQL result map is the result of a TQL calculation. It contains data according to node and links numbers defined in the TQL pattern definition. The map contains the `CHUNKED` flag, the `CHUNKSINFO` chunk information, the `OBJECTSRESULTENTRIES` objects results list, and the `LINKSRESULTENTRIES` links results list.

The `CHUNKED` flag is true if the result map is large and should be retrieved in chunks. If so, only the chunk information is used in processing the map. If the flag is false, the data is in the objects results list and links results list of this result map.

The chunk information is sent as the request identifier in each chunk request if `CHUNKED` is true.

The entries in the objects result entries list contain a node number defined in the TQL definition and a list of the objects that belong to this node.

The entries in the links result entries contain a link node number defined in the TQL definition and a list of the links that belong to this link node.

Chunking is never more than two layers deep. One map that is chunked has its information passed in the objects results and links results lists of a number of partial maps, none of which is chunked itself.

## **CMDB Module Methods**

This section includes the following topics:

- CMDB Module Update Methods. For details, see below.
- CMDB Module Query Methods. For details, see page 47.

### **CMDB Module Update Methods**

#### **modelUpdateAddOrUpdateData**

`modelUpdateAddOrUpdateData` adds or updates objects and links in the data container. If objects or links do not exist in the container, they are added. If they already exist, they are updated with the new data.

Input:

- cmdb-context
- data-container

Output:

- Map. A map of key strings and value strings. The key is a temporary ID and the value is the real CMDB ID. The temporary ID is known to the client because it is initially generated by the client. The Map can be used to get the corresponding real CMDB ID.

### **modelUpdateUpdateData**

`modelUpdateUpdateData` updates the objects and links that are in the data container. If any of the objects or links do not exist in the CMDB, an exception is thrown.

Input:

- cmdb-context
- data-container

### **modelUpdateRemoveDataIfExist**

`modelUpdateRemoveDataIfExist` removes the objects and links that are in the data container. If any of the objects or links do not exist in the CMDB, an exception is thrown.

Input:

- cmdb-context
- data-container

## CMDB Module Query Methods

### **modelQueryCalcTqlByDefinition**

modelQueryCalcTqlByDefinition performs an ad-hoc calculation of the specified TQL and returns a TQL result map.

Input:

- cmdb-context
- tqlDefinition string. The CMDB graph XML defines the TQL.

Output:

- tql-result-map

### **modelQueryGetCIsByCondition**

modelQueryGetCIsByCondition returns a tql-result-map of all data objects that match the specified conditions. modelQueryGetCIsByCondition returns only objects, not links.

Input:

- cmdb-context
- elementCondition. The condition to be matched in order to return CIs. An element-condition consists of three parameters. All parameters specified must be true for the condition to be met. The parameters are:
  - classCondition. The name of the class for which to retrieve CIs. If parameter ISDERIVED is true, CIs of all descendant classes are also returned. Otherwise only CIs of given class will be returned. This parameter is required.
  - idsCondition. The CMDB ids of the CIs to be retrieved. This parameter is optional.
  - propertyConditions. Specifications of conditions based on CIs' properties. All property conditions must be met for this parameter to be true. This parameter is optional.

- ▶ `elementTypeLayout`. The layout of CIs to retrieve.

Output:

- ▶ `tql-result-map`. The matching objects are returned in node number 1.

### **modelQueryGetTqlResultByTqlName**

`modelQueryGetTqlResultByTqlName` returns the existing result of the specified active TQL, or performs an ad hoc calculation of any TQL in the CMDB.

Input:

- ▶ `cmdb-context`
- ▶ `tqlName`. The name of the TQL in the CMDB for which to get the result
- ▶ `isToRunAdHoc` (boolean). Sets whether to use existing results or force an ad hoc run. If the value is **true**, the TQL is run ad hoc whether it is active or not. If false, the existing results of the active TQL are used.

Output:

- ▶ `tql-result-map`

### **modelQueryGetTqlResultChunk**

`modelQueryGetTqlResultChunk` retrieves a chunk of a `tql-result-map`. Use this method when `ISCHUNKED` is true in the result map returned by another method.

Input:

- ▶ `cmdb-context`
- ▶ `tql-chunk-request`. The request has an integer index named `CHUNKTORETRIEVE`, and `CHUNKINFO`. The client should not create this `CHUNKINFO`, but use the chunk information returned in the `tql-result-map` returned by methods like `modelQueryGetTqlResultByTqlName` or `modelQueryCalcTqlByDefinition`.



Output:

- The `tql-result-map` of the requested chunk. This map is never chunked, so the `isChunked` flag and chunk information should be ignored. Only the objects results entries and the links results entries contain useful information.

### **modelQueryGetDirectAssociations**

`modelQueryGetDirectAssociations` returns the neighbors of the specified CMDB source object.

Input:

- `cmdb-context`
- `wsSourceCmdbObject`. The source CMDB data object for which to return the direct associations.
- `depth`. The maximum depth of neighbors in the `cmdb-graph` to return.
- `wsDataLayout`. A data-layout for the results. The result contains the objects and links that are direct associates of the source object.
- The `wsTqlPatternInfo`. The `tql-pattern-info` is a filter for associations that fit the pattern or a specified node in the pattern. If the direct associations should be retrieved over the whole CMDB model, set this parameter to null. The TQL Pattern Information consists of:

`tqlName`. The name of the TQL from which to retrieve direct associations.

`nodeNumber` (integer). The node number in the specified TQL where the source object is located.

Output:

- `tql-result-map`. Objects are returned in node number 1 and links are returned in node number 2. The map can be chunked.

### **modelQueryGetDirectDependencies**

`modelQueryGetDirectDependencies` returns a `cmdb-graph` of all CIs connected directly to the given object by links of type `CONTAINER` or type `CONTAINED`.

Input:

- ▶ `cmdb-context`
- ▶ The source CMDB data object. The object whose dependencies to return.
- ▶ `data-layout`. Defines the layout of the result to return. The result contains the objects and links that are direct dependencies of the source object.

Output:

The `cmdb-graph` of all direct dependencies of the source object.

## Use Cases and Examples

There are two categories of use, updating the CMDB and querying the CMDB. The following examples assume two systems:

Mercury CMDB server

A third party system that contains a repository of CIs (configuration items).

### Population of the Mercury CMDB

A third party system can update the Mercury CMDB by:

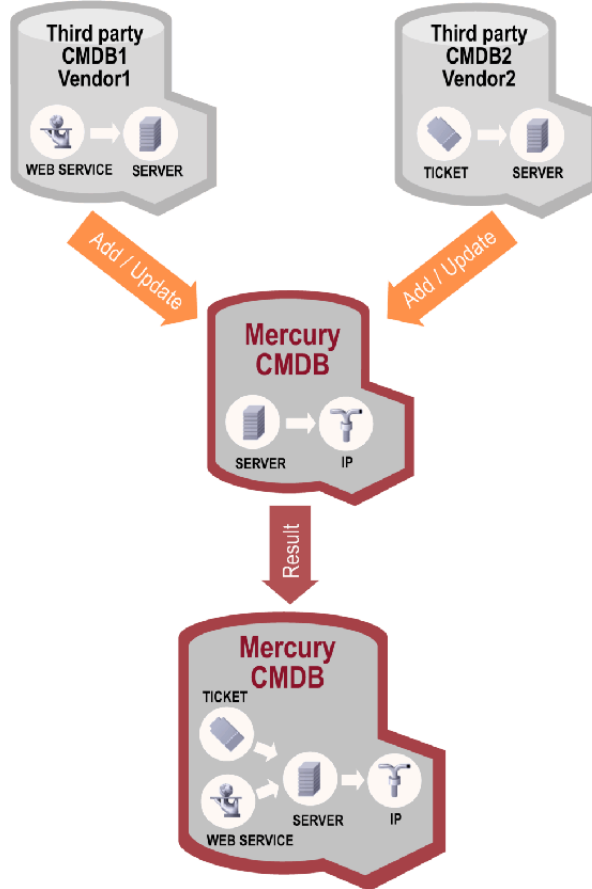
- ▶ Populating the Mercury CMDB with new CIs and Relations based on the class definition already defined in Mercury CMDB
- ▶ Removing CIs and Relations from the Mercury CMDB
- ▶ Updating existing CIs and Relations in the Mercury CMDB

Use cases:

- ▶ A third party asset management updates the Mercury CMDB with information available only in asset management
- ▶ A number of third party systems populate the Mercury CMDB to create a central CMDB that can track changes and perform impact analysis
- ▶ A third party system creates CIs and Relations according the third party business logic to leverage the CMDB query capabilities

## Update use case diagram

Third party asset management systems update the Mercury CMDB with information available only in asset management.



After the update process is complete, the CMDB can be queried and you can discover that there is a ticket that may affect a web service.

## Update Code Example

```
package demo;

import client.generated.*;
import client.generated.Object;

import javax.xml.rpc.ServiceException;
import java.rmi.RemoteException;

/**
 * *****
 * The CmdbUpdateApiSimpleDemo class is an example
 * of updating the CMDB.
 * *****
 */
public class CmdbUpdateApiSimpleDemo {

    private int _customerID = 1;
    private String _callerApp = "CmdbUpdateApiSimpleDemo";

    /**
     * This example adds a document, then creates a
     * link between a host that is already
     * in the CMDB and this new document.
     * @param hostKey    The key of the host
     * @throws           ServiceException
     * @throws           RemoteException
     */
    public void add(String hostKey) throws
        ServiceException, RemoteException {

        // Create the data container.
        DataContainer dataContainer = new DataContainer();

        // Create the reference to the host that is
        // already in the CMDB.
        String hostID = "ref1";
        Object host = new Object();
```

```

host.setType("host");
host.setID(hostID);
host.setIsCmdbID(false);

// Set the properties of the host.
StrProp hostKeyProp = new StrProp();
hostKeyProp.setK("host_key");
hostKeyProp.setV(hostKey);
StrProp[] stringHostProps = {hostKeyProp};
Props hostProps = new Props();
hostProps.setStrProps(stringHostProps);
host.setProps(hostProps);

Object[] hostArr = {host};
Objects refObjects = new Objects(hostArr);
dataContainer.setReferencedObjects(refObjects);

// Create the document.
String docID = "ID2";
Object document = new Object();
document.setType("document");
document.setID(docID);
document.setIsCmdbID(false);

// Set the properties of the document and add
// it to the dataContainer.
StrProp documentNameProp = new StrProp();
documentNameProp.setK("data_name");
documentNameProp.setV("doc_"+hostKey);
StrProp documentContainerProp = new StrProp();
documentContainerProp.setK("root_container");
documentContainerProp.setV(hostID);
StrProp[] stringDocProps =
    {documentNameProp,documentContainerProp};
Props docProps = new Props();
docProps.setStrProps(stringDocProps);
document.setProps(docProps);
Object[] docArr = {document};

```

```
Objects objects = new Objects(docArr);
dataContainer.setCmdbObjects(objects);

// Create the link and add it to the data container.
Link link = new Link();
link.setType("container_f");
link.setID("ID3");
link.setEnd1ID(hostID);
link.setEnd1IDIsCmdbID(false);
link.setEnd2ID(docID);
link.setEnd2IDIsCmdbID(false);
Link[] linkArr = {link};
Links links = new Links(linkArr);
dataContainer.setCmdbLinks(links);

// Update the CMDB.
getCmdbOpenApi().modelUpdateAddOrUpdateData
    (getCmdbContext(),dataContainer);
}

/**
 * This example updates the host's DNS name property.
 * @param hostKey The key of the host
 * @throws ServiceException
 * @throws RemoteException
 */
public void update(String hostKey) throws
    ServiceException, RemoteException {

// Create the data container.
DataContainer dataContainer = new DataContainer();

// Create the host object.
Object host = new Object();
host.setType("host");
host.setID("ID1");
host.setIsCmdbID(false);
```

```

// Create the key properties because
// the real CMDB ID is unknown.
StrProp hostKeyProp = new StrProp();
hostKeyProp.setK("host_key"); // Property "host_key"
hostKeyProp.setV(hostKey);

// Create the property to be updated and set
// the new value.
StrProp hostDnsProp = new StrProp();
hostDnsProp.setK("host_dnsname");
hostDnsProp.setV("newDnsName");

// Create the collection of properties and add
// it to the host object.
StrProp[] hostStringProps = {hostKeyProp,hostDnsProp};
Props hostProps = new Props();
hostProps.setStrProps(hostStringProps);
host.setProps(hostProps);

// Add the host to the data container.
Object[] objectsArray = {host};
dataContainer.setCmdbObjects(new Objects(objectsArray));

// Update the CMDB.
getCmdbOpenApi().modelUpdateUpdateData
    (getCmdbContext(),dataContainer);
}

/**
 * This example removes the specified host.
 * @param hostKey The key of the host
 * @throws ServiceException
 * @throws RemoteException
 */
public void remove(String hostKey)
    throws ServiceException, RemoteException {

// Create the data container
DataContainer dataContainer = new DataContainer();

```

```
// Create the host object
Object host = new Object();
host.setType("host");
host.setID("ID1");
host.setIsCmdbID(false);

// Create the key properties because
// the real CMDB ID is unknown.
StrProp hostKeyProp = new StrProp();
hostKeyProp.setK("host_key");
hostKeyProp.setV(hostKey);

// Create the collection of properties and add
// it to the host object.
StrProp[] hostStringProps = {hostKeyProp};
Props hostProps = new Props();
hostProps.setStrProps(hostStringProps);
host.setProps(hostProps);

// Add the host to the data container.
Object[] objectsArray = {host};
dataContainer.setCmdbObjects(new Objects(objectsArray));

// Update the CMDB.
getCmdbOpenApi().modelUpdateRemoveDataIfExist
    (getCmdbContext(),dataContainer);
}

private CmdbOpenApi getCmdbOpenApi()
    throws ServiceException {

// Create the CMDB API.
CmdbOpenApiServiceLocator cmdbOpenApiServiceLocator =
    new CmdbOpenApiServiceLocator();
return cmdbOpenApiServiceLocator.getCmdb();
}
```



```
private CmdbContext getCmdbContext(){
    return new CmdbContext(_callerApp,_customerID);
}
}
```

## Querying the Mercury CMDB

There are many approaches to queries. For example:

The client gets the result of a TQL that is defined in the CMDB or passes a TQL definition to CMDB. This approach is used for a complex queries, for example, topology queries.

For an ad hoc query, the calling party provides a TQL. Usually, the calling party exports a TQL definition and then uses it in the ad hoc query.

For simple queries where topology is not important, a single CI type is queried based on a condition. This type of query returns a list of CIs, rather than a graph.

The topology surrounding a specific CI is queried using APIs, for example, `modelQueryGetDirectAssociations` or `modelQueryGetDirectDependencies`. This information can also be extracted with a TQL query.

The CMDB can be queried for all the neighbors of a given CI in a specified range. For example, the query can be performed over a specific TQL that returns all neighbors in context of a SAP application, or a TQL that returns all neighbors in the CMDB.

The CMDB can be queried for direct decedents, that is, items that are contained by the CI. For example, querying a host CI returns the host's disks.

Use cases:

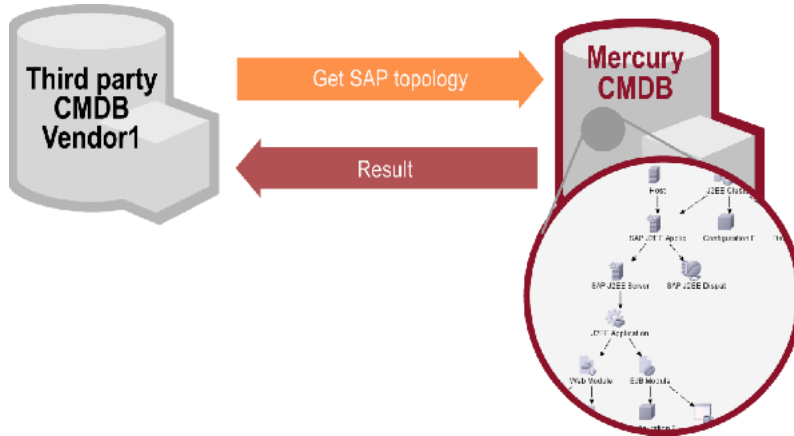
A third party system gets the CIs and Relations that represent the SAP system by getting the results of the SAP TQL.

A third party system gets the list of Oracle servers that have been added or changed in the last five hours.

A third party system gets the list of servers whose host name contains the substring “lab”.

A third party system finds the relations to a given CI by getting its neighbors.

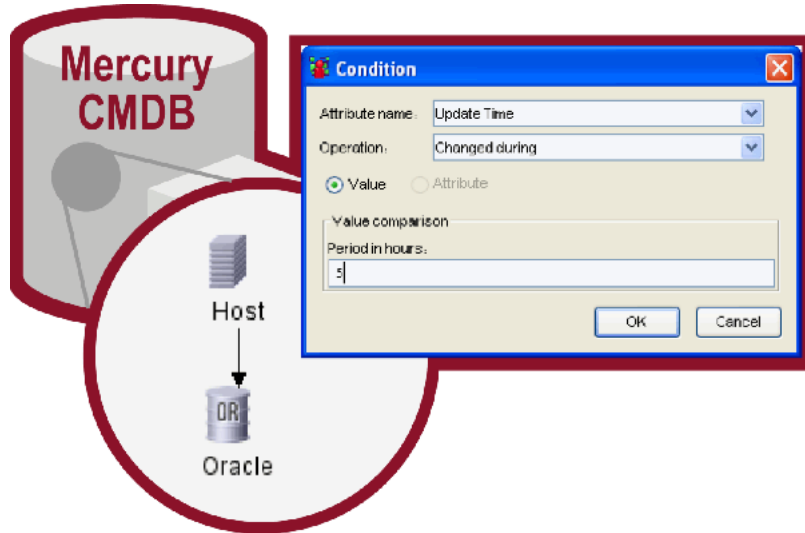
### Getting all CIs and Relations from SAP



### Finding Changed Oracle Servers

A third party system that monitors servers running the Oracle database manager gets a list of the servers running Oracle.exe that had been added or changed in the last five hours by running an ad hoc query.

Create a TQL and export it to XML.

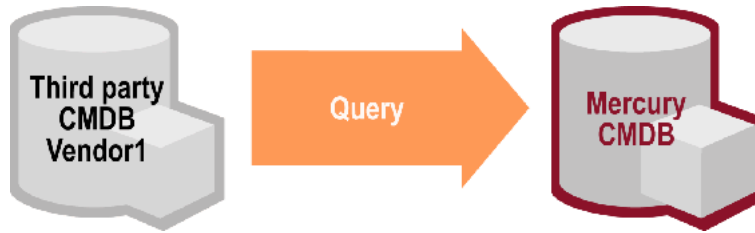


A third party application runs the query based on the exported XML.



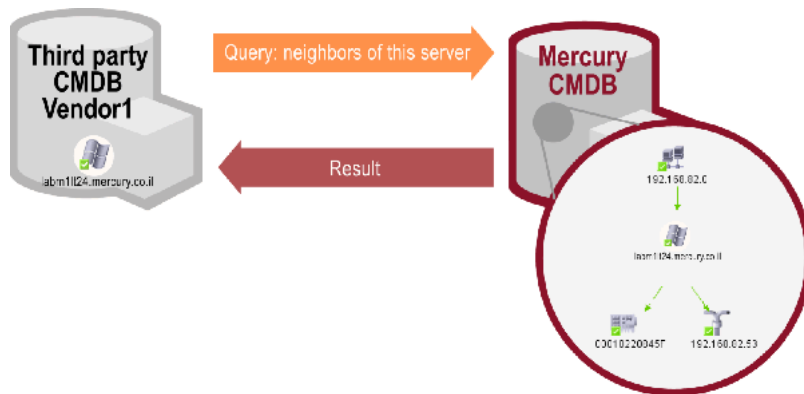
### Finding hosts with a host name LIKE %Lab%

A third party system gets a list of servers that have a host name containing the substring “lab” by performing a condition-based query based host type and host name.



### Finding relations of CIs to a given CI.

A third party system containing CIs queries the CMDB on relations of a given CI to other CIs by performing a get-neighbors query on the CMDB.



### Query Code Example

```
package demo;  
  
import client.generated.*;  
import client.generated.Object;  
  
import javax.xml.rpc.ServiceException;
```

```

import java.rmi.RemoteException;

/**
 * *****
 * The CmdbContextQueryApiSimpleDemo class is an
 * example of getting data from the CMDB.
 * *****
 */
public class CmdbContextQueryApiSimpleDemo {

    private int _customerID = 1;
    private String _callerApp = "CmdbContextQueryApiSimpleDemo";

    /**
     * Runs the specified TQL and prints the resulting map.
     * @throws ServiceException
     * @throws RemoteException
     */
    public void getTqlResultByTqlDefinition(String tqlDefinition)
        throws ServiceException, RemoteException {

        CmdbContextOpenApi api = getCmdbContextOpenApi();
        CmdbContext cmddbContext = getCmdbContext();

        // Run the query in the CMDB.
        TqlResultMap tqlResultMap =
            api.modelQueryCalcTqlByDefinition(cmddbContext,tqlDefinition);

        // Print the results.
        printAllTqlResultMapChunks(tqlResultMap,api,cmddbContext);

    }

    /**
     * Runs the "Host Resources" TQL and prints the
     * resulting map.
     * @throws RemoteException
     * @throws ServiceException
     */
}

```

```
public void getTqlResultByTqlName()
    throws RemoteException, ServiceException {

    CmdbOpenApi api = getCmdbOpenApi();
    CmdbContext cmdbContext = getCmdbContext();
    String tqlname = "Host Resources";

    // Run the query in the CMDB
    TqlResultMap tqlResultMap =
        api.modelQueryGetTqlResultByTqlName(cmdbContext, tqlname,false);

    // Print the results
    printAllTqlResultMapChunks(tqlResultMap,api,cmdbContext);
}

/**
 * Gets all the neighbors of the host and prints
 * the resulting map.
 * @param hostKey    The host's key
 * @throws           ServiceException
 * @throws           RemoteException
 */
public void getDirectAssociations(String hostKey)
    throws ServiceException, RemoteException {

    CmdbOpenApi api = getCmdbOpenApi();
    CmdbContext cmdbContext = getCmdbContext();

    // Create the host object
    // Because the real CMDB ID is not known,
    // sends ID properties.
    Object host = new Object();
    host.setType("host");
    host.setIsCmdbID(false);
    StrProp hostKeyProp = new StrProp();
    hostKeyProp.setK("host_key");
    hostKeyProp.setV(hostKey);
    StrProp[] hostStringProps = {hostKeyProp};
    Props hostProps = new Props();
```

```

hostProps.setStrProps(hostStringProps);
host.setProps(hostProps);

// Create the layout
ElementTypeLayout elementTypeLayout =
    new ElementTypeLayout();
elementTypeLayout.setKeys
    (new String[]{"data_name","display_label"});
DataLayout dataLayout =
    new DataLayout(elementTypeLayout,elementTypeLayout);

// Run the query over the entire CMDB model and
// not over a pattern result.
TqlResultMap tqlResultMap =
    api.modelQueryGetDirectAssociations
        (cmdbContext, host,2,dataLayout,null);
printTqlResultMap(tqlResultMap);
}

/**
 * Gets all the direct dependencies of the host
 * and outputs the objects and links.
 * @param hostKey The host's key
 * @throws ServiceException
 * @throws RemoteException
 */
public void getDirectDependencies(String hostRealCmdbID)
    throws ServiceException, RemoteException {

    CmdbOpenApi api = getCmdbOpenApi();
    CmdbContext cmdbContext = getCmdbContext();

    // Create the host object.
    Object host = new Object();
    host.setType("host");
    host.setID(hostRealCmdbID);
    host.setIsCmdbID(true);

    // Create the layout.

```

```
ElementTypeLayout elementTypeLayout =
    new ElementTypeLayout();
elementTypeLayout.setKeys
    (new String[]{"data_name","display_label"});
DataLayout dataLayout =
    new DataLayout(elementTypeLayout,elementTypeLayout);

// Run the query.
CmdbGraph cmdbGraph =
    api.modelQueryGetDirectDependencies
        (cmdbContext, host, dataLayout);
System.out.println("found objects: "+cmdbGraph.getCmdbObjects());
System.out.println("found links: "+cmdbGraph.getCmdbLinks());
}

/**
 * Gets all Configuration Items whose type is either
 * host or a type derived from host
 * and prints the resulting map.
 * @throws RemoteException
 * @throws ServiceException
 */
public void getCIsByClassConditionOnly()
    throws RemoteException, ServiceException {

    CmdbOpenApi api = getCmdbOpenApi();
    CmdbContext cmdbContext = getCmdbContext();

    // Create the class condition: host or derived from host.
    ClassCondition classCondition =
        new ClassCondition("host", Boolean.TRUE);

    // Build the element condition. It includes only
    // the class condition.
    ElementCondition elementCondition =
        new ElementCondition(classCondition, null, null);

    // Run the query and print the results.
    TqlResultMap tqlResultMap =
```



```

    api.modelQueryGetClsByCondition
        (cmdbContext, elementCondition,
         getElementTypeLayoutForObjects());
    printAllTqIResultMapChunks(tqIResultMap,api,cmdbContext);
}

/**
 * Gets all Configuration Items whose ID is in
 * the input array and
 * whose type is either
 * host or a type derived from host,
 * and prints the resulting map.
 * @param cmdbIDs - array of CMDB IDs
 * @throws ServiceException
 * @throws RemoteException
 */
public void getClsByClassConditionAndIdsCondition(String[] cmdbIDs)
    throws ServiceException, RemoteException {

    CmdbOpenApi api = getCmdbOpenApi();
    CmdbContext cmdbContext = getCmdbContext();

    // Create the class condition: host or derived from host.
    ClassCondition classCondition =
        new ClassCondition("host", Boolean.TRUE);

    // Create the IdsCondition according to the input.
    IdsCondition idsCondition = new IdsCondition(cmdbIDs);

    // Build the element condition. It includes the class
    // condition and the IDs condition.
    ElementCondition elementCondition =
        new ElementCondition(classCondition, idsCondition, null);

    // Run the query and print the result.
    TqIResultMap tqIResultMap =
        api.modelQueryGetClsByCondition
            (cmdbContext, elementCondition,
             getElementTypeLayoutForObjects());

```

```

    printAllTqIResultMapChunks(tqIResultMap,api,cmdbContext);
}

/**
 * Gets all Configuration Items whose type is either host
 * or a type derived from host and which have the specified
 * DNS name property, prints the resulting map.
 * @throws ServiceException
 * @throws RemoteException
 */
public void getCIsByClassConditionAndPropertyCondition
(String dnsName)
throws ServiceException, RemoteException {

    CmdbOpenApi api = getCmdbOpenApi();
    CmdbContext cmdbContext = getCmdbContext();

    // Create the class condition: host or derived from host.
    ClassCondition classCondition =
        new ClassCondition("host", Boolean.TRUE);

    // Create the property condition with EqualOperator.
    StrPropCondition dnsNameCondition =
        new StrPropCondition();
    dnsNameCondition.setPropName("host_dnsname");
    dnsNameCondition.setConditionOperator(new EqualOperator());
    dnsNameCondition.setValue(dnsName);
    PropConditions propConditions =
        new PropConditions(new PropCondition[]{dnsNameCondition});

    // Build the element condition. It includes the class
    // condition and the property condition.
    ElementCondition elementCondition =
        new ElementCondition(classCondition, null, propConditions);

    //Run the query and print the results
    TqIResultMap tqIResultMap =
        api.modelQueryGetCIsByCondition
            (cmdbContext, elementCondition,

```

```

        getElementLayoutForObjects());
    printAllTqlResultMapChunks(tqlResultMap,api,cmdbContext);
}

/**
 * Gets all chunks in the TqlResultMap and send
 * them to be printed.
 * @param tqlResultMap
 * @param api
 * @param cmdbContext
 * @throws RemoteException
 */
private void printAllTqlResultMapChunks
    (TqlResultMap tqlResultMap, CmdbOpenApi api,
     CmdbContext cmdbContext)
    throws RemoteException {

    //If chunked, send each individual chunk to be printed.
    if (tqlResultMap.isChunked()) {
        ChunksInfo chunksInfo = tqlResultMap.getChunksInfo();
        int chunkNum = chunksInfo.getChunksNum();
        for (int i = 0; i < chunkNum; i++) {

            //Get each chunk and send its map to be printed.
            TqlChunkRequest tqlChunkRequest =
                new TqlChunkRequest();
            tqlChunkRequest.setChunkInfo(chunksInfo);
            tqlChunkRequest.setChunkToRetrieve(i);
            tqlResultMap =
                api.modelQueryGetTqlResultChunk
                    (cmdbContext, tqlChunkRequest);
            printTqlResultMap(tqlResultMap);
        }
    }
    //If not chunked, send the map as-is to be printed.
    else {
        printTqlResultMap(tqlResultMap);
    }
}

```

```

/**
 * Prints a TqlResultMap.
 * @param tqlResultMap
 */
private static void printTqlResultMap(TqlResultMap tqlResultMap){

    ObjectsResultEntry[] objectsResultEntries =
        tqlResultMap.getObjectsResultEntries();
    for (int i=0;i<objectsResultEntries.length;i++){
        System.out.println
            ("TQL object node number:
             "+objectsResultEntries[i].getElementNum());
        System.out.println
            ("Objects in the node:
             "+objectsResultEntries[i].getObjects());
    }
    LinksResultEntry[] linksResultEntries =
        tqlResultMap.getLinksResultEntries();
    for (int i=0;i<linksResultEntries.length;i++){
        System.out.println
            ("TQL link node number:
             "+linksResultEntries[i].getElementNum());
        System.out.println
            ("Links in the node:
             "+linksResultEntries[i].getLinks());
    }
}

/**
 * Returns the layout for objects.
 * @return ElementTypeLayout The layout for objects.
 */
private static ElementTypeLayout getElementTypeLayoutForObjects() {
    ElementTypeLayout elementTypeLayout =
        new ElementTypeLayout();
    elementTypeLayout.setKeys
        (new String[]{"data_name", "display_label"});
}

```

```

        return elementTypeLayout;
    }

    /**
     * Returns CmdbOpenApi
     * @return CmdbOpenApi
     * @throws ServiceException
     */
    private CmdbOpenApi getCmdbOpenApi() throws ServiceException {
        // Create the CMDB API
        CmdbOpenApiServiceLocator cmdbOpenApiServiceLocator =
            new CmdbOpenApiServiceLocator();
        return cmdbOpenApiServiceLocator.getCmdb();
    }

    /**
     * Returns the CmdbContext according to customer
     * ID and caller application.
     * @return CmdbContext
     */
    private CmdbContext getCmdbContext(){
        return new CmdbContext(_callerApp,_customerID);
    }
}

```



# **Part III**

---

## **Working with Mercury Business Availability Center EMS Adapters**





# 6

---

## Introducing Mercury Business Availability Center EMS Adapters

EMS adapters enable you to integrate alerts generated by Mercury Business Availability Center into your EMS console. These alerts can then be used in the standard alert-processing procedure used in the EMS application.

---

**Note to Mercury Managed Services customers:** For details on acquiring EMS adapters, contact Mercury Managed Services Support.

---

You can send Mercury Business Availability Center alerts to EMS applications with the following adapters:

- ▶ **BMC PATROL Adapter.** For details, see Chapter 7, “Sending Mercury Business Availability Center Alerts to BMC PATROL.”
- ▶ **CA Unicenter Adapter.** For details, see Chapter 8, “Sending Mercury Business Availability Center Alerts to CA Unicenter.”
- ▶ **HP Openview VantagePoint Operations (OVO) Adapter.** For details, see Chapter 9, “Sending Mercury Business Availability Center Alerts to HP Openview VantagePoint Operations.”
- ▶ **IBM Tivoli Enterprise Console (TEC) Adapter.** For details, see Chapter 10, “Sending Mercury Business Availability Center Alerts to Tivoli TEC.”

---

**Note:** For details on assigning alerts in Mercury Business Availability Center, see “How Do I Set Up a Report and Alert Delivery System?” in *Getting Started with Mercury Business Availability Center*.

---



# 7

---

## **Sending Mercury Business Availability Center Alerts to BMC PATROL**

This chapter explains how to configure Mercury Business Availability Center to send alerts to BMC PATROL.

<b>This chapter describes:</b>	<b>On page:</b>
About Sending Mercury Business Availability Center Alerts to BMC PATROL	75
Installing the Mercury Business Availability Center SNMP Adapter on the BMC PATROL Agent(s)	76
Installation Troubleshooting	76

### **About Sending Mercury Business Availability Center Alerts to BMC PATROL**

You install the Mercury Business Availability Center SNMP Adapter on the BMC PATROL Agent(s) to which you want to send alerts. You then direct SNMP traps generated by the Core Server to the BMC PATROL Agent(s). For details on directing SNMP traps, see “Sending an SNMP Trap When the Alert Is Triggered” on page 245.

You do not have to install the SNMP Adapter on every BMC PATROL Agent in the BMC PATROL domain: a single agent per domain should suffice. However, if required, you can install the SNMP Adapter on several agents, to achieve higher availability.

## Installing the Mercury Business Availability Center SNMP Adapter on the BMC PATROL Agent(s)

Once installed, the SNMP Adapter receives alerts (in the form of SNMP traps) from Mercury Business Availability Center and forwards them to the event management engine of the BMC PATROL Agent(s) on which the SNMP Adapter operates.

**To install the Mercury Business Availability Center adapter on BMC PATROL:**

- 1 Copy the Mercury Business Availability Center SNMP Adapter Knowledge Module (TOPAZ\_SNMP\_ADAPTER.km) to the knowledge module location of both the BMC PATROL Console and BMC PATROL Agent(s).

TOPAZ\_SNMP\_ADAPTER.km is included in the Adapter files. The file is located on the Mercury Business Availability Center Documentation and Utilities CD-ROM in the \tools\_and\_utilities\EMS\BmcPatrol\TSA folder. Once the knowledge module is loaded into BMC PATROL, TOPAZ\_SNMP\_ADAPTER.km is accessible via the BMC PATROL developer console.

- 2 Load the Mercury Business Availability Center SNMP Adapter on the BMC PATROL Agent(s) using the BMC PATROL Developer Console. For exact details, refer to the BMC PATROL documentation.

## Installation Troubleshooting

**BMC PATROL Agent(s) output(s) the following system message (number(s) on the left may vary):  
20040502153419 SNMP NOT ACTIVE.**

The SNMP Trap service is not enabled on the BMC PATROL Agent on which the SNMP Adapter is installed. Enable the SNMP Trap service on that agent, or use a different BMC PATROL Agent. Refer to the BMC PATROL documentation for details on installing and enabling the SNMP trap service on BMC PATROL Agents.

**BMC PATROL Agent(s) output(s) the following system message (number(s) on the left may vary):**

**20040502154044 SnmpListener, Line# 130:**

**snmp\_trap\_listen:**

**cannot open socket for listening**

**20040502154044 Listen status = ERR, Errno = 94.**

Another SNMP trap service is using the SNMP trap port. Disable that service or change the UDP port that is used for trap reception (in that case, you must reconfigure Mercury Business Availability Center).

**BMC PATROL Agent(s) output(s) the following system message (number(s) on the left may vary):**

**20040502154044 Error: Can't recognize Alert Type Id X (where X is a number).**

The Mercury Business Availability Center SNMP Adapter receives an SNMP trap that is not recognized as a valid Mercury Business Availability Center alert. This is normal behavior.

---

**Note:** For testing purposes, you may want to enable auxiliary SNMP traps to be processed. This can be achieved by changing the KM environment variable **RAW\_DATA** to **TRUE**.

---



# 8

---

## Sending Mercury Business Availability Center Alerts to CA Unicenter

This chapter explains how to configure Mercury Business Availability Center to send alerts to CA Unicenter.

This chapter describes:	On page:
About Sending Mercury Business Availability Center Alerts to CA Unicenter	79
Sending a Mercury Business Availability Center Alert as an SNMP Trap	80
Sending a Mercury Business Availability Center Alert Using the Unicenter cawto Command	81
Configuration Troubleshooting	83

### About Sending Mercury Business Availability Center Alerts to CA Unicenter

To send alerts to the CA Unicenter application, you can use one of the following methods:

- ▶ Sending a Mercury Business Availability Center Alert as an SNMP Trap to the Unicenter Management server
- ▶ Sending a Mercury Business Availability Center Alert Using the Unicenter cawto Command

For details on directing SNMP traps, see “Sending an SNMP Trap When the Alert Is Triggered” in *Platform Administration*.

## Sending a Mercury Business Availability Center Alert as an SNMP Trap

The following procedure explains how to configure Unicenter to pick up the SNMP traps sent to the Event Management host.

### To configure Unicenter:

- 1 Enable the Unicenter **CATrapD** (SNMP trap daemon) to pick up the raw SNMP traps.
- 2 Configure Event Management (Message Records and Actions) to format the traps according to the Mercury Business Availability Center MIB. This requires knowledge of the Mercury Business Availability Center MIB that is part of the Core Server installation. The file resides on the Mercury Business Availability Center Documentation and Utilities CD-ROM in the `\tools_and_utilities\SNMP_MIBS\amAlerts5.mib` folder.

### To configure the Management Server host:

You must create a message record to fit the SNMP trap sent by Mercury Business Availability Center.

- 1 Create a `[file name].def` file and save the following line in it:

```
define msgrec msgid="%CATD_I_060, SNMPTRAP: -c * 5233 * * 6 1 * *"  
type="MSG" msgnode="*" desc="AM Alert Transaction Response Time"  
cont='N' msgact='Y' wcsingle='?' wcmay='*' case="y" regexp="n"
```

- 2 Load the file as part of the message records in Event Management by using the command:

```
cautil -f [file name].def  
opr cmd opreload
```

- 3 To receive and display SNMP traps from third party devices in the Unicenter Event Console, you must enable the SNMP Trap Service. Select **Unicenter Enterprise Management > Configuration > Settings**. Change the **SNMP Trap Server Activated** flag to **Yes**.



**To configure Mercury Business Availability Center:**

- 1** To configure the alerts that are to be sent to the Unicenter Event Management host, access the Alert Wizard: Select **Admin > Platform > Alerts > Alerts**.
- 2** Create an alert according to the instructions in “Creating an Alert Scheme” on page 198.
- 3** Select the **Send SNMP trap** check box and define the EMS IPs. For details, see “Sending an SNMP Trap When the Alert Is Triggered” on page 245.

## **Sending a Mercury Business Availability Center Alert Using the Unicenter `cawto` Command**

---

**Note to Mercury Managed Services customers:** This procedure is not relevant for Mercury Managed Services customers. Instead, Mercury Managed Services customers should use the SiteScope command line monitor. To use that monitor, consult Mercury Managed Services Support.

---

This command line utility is available after you install the Unicenter Event Management Agent on the Mercury Business Availability Center machine.

---

**Note:** You use the `cawto` command to send a message to the Windows console or the system console without waiting for a reply. To send a message and wait for a reply, use the Event Management `cawtor` command.

---

**To configure the CA Unicenter Console:**

You do not need to configure the Unicenter Event Management host. Verify that the `cawto` command is sent to the Unicenter Event Management node which must handle it: open the Event Management Console and look for the event that was sent with the `cawto` command.

**To configure Mercury Business Availability Center:**

- 1** To configure the alerts that are to be sent to the Unicenter Event Management host, access the Alert Wizard: Select **Admin > Platform > Alerts and Recipients > Alerts**.
- 2** Create an alert according to the instructions in “Creating an Alert Scheme” in *Platform Administration*.
- 3** Continue to the section “Running an Executable File When the Alert is Triggered” in *Platform Administration*. During this procedure, you must select the **Run executable file** check box and define the alert action in the Alert Wizard. When asked to enter the command line required to run the executable file, use the Unicenter **cawto** command line.

---

**Note:** Consult the Unicenter documentation about the parameters for the **cawto** command line and how they can be activated.

---

The following example shows how to create a **cawto** command that sends the Mercury Business Availability Center alert to Unicenter:

```
C:\tng\bin\cawto.exe -s "" -v W -n <node> "Alert <AlertName> for <ProfileName> was triggered because: <TriggerCause>"
```

where:

**-s <source>** – Identifies the application that is the source of the event. In the above example, **<source>** is **Mercury Business Availability Center**.

**-v <value>** – Severity of the event. When viewing events in the CA Message console, icons are displayed to the left of the events indicating the severity status. In the above example, **<value>** is **W** (Warning). Severity values can be one of the following:

Informational	I
Success	S
Warning	W

Error	E
Failure	F

You can set up the translation from Mercury Business Availability Center severity to Unicenter severity according to the following rules:

	Unicenter
Informational	I
Warning	W
Minor	E
Major	E
Critical	F

**-n <node>** – Node to which the message is directed if the node is not the node the user is on.

**<AlertName>**, **<ProfileName>**, **<TriggerCause>** are Mercury Business Availability Center alert variables.

## Configuration Troubleshooting

**I have a custom or third party MIB. What exactly will loading it on the DSM, using the LDMIB command, enable me to do?**

**Products:**

- Unicenter NSM

Version: 3.0

OS: WIN/NT, WIN2000, HP, AIX, SUN, LINUX, SCO, SINIX, TRU64

- Unicenter TNG

Version: 2.2, 2.4, 2.4.2

OS: WIN/NT, WIN2000, HP, AIX, SUN, LINUX, SCO, SINIX, TRU64

**Solution:**

Run **ldmib** on a custom MIB to give the following functionality:

- ▶ the ability to run **objectview** against that MIB. This requires that the MIB also be copied into the **%AGENTWORKS\_DIR\services\config\mibs** directory on the machine from which **objview** is being run.
- ▶ the ability to run **mibbrowse** against that MIB.

Please note that no trap translation automatically occurs by loading an MIB. This requires DSM policy. You could use message records and actions to translate the **CATrapD** messages on the EM console to work around the need for DSM policy. This would then require enabling **CATrapD** by accessing the EM settings and turning on **SNMP trap server activated**.

**I have been receiving many messages on the event console that have an unreadable format, such as '%CATD\_I\_060, SNMPTRAP: -c public 791 172.20.0.18 mail.npc.net 2 0.' What are they?**

**Products:**

- ▶ Unicenter NSM  
Version: 3.0  
OS: WIN/NT, WIN2000
- ▶ Unicenter TNG  
Version: 2.2, 2.4, 2.4.2  
OS: WIN/NT, WIN2000

**Solution:**

These are raw SNMP traps that the **CATrapD** daemon picks up. They may be generated by any SNMP enabled device or agent. **CATrapD** is enabled when you access the EM settings and enable **SNMP trap server activated**. Normally the DSM interprets the important ones but others do not warrant a separate translated message in the console. The reference guide lists the numbers at the end of these traps and what they stand for.

They are:

0	Coldstart	The sending SNMP entity has reinitialized itself, indicating that the agent's configuration may be changed. This is typically a restart due to a crash or major fault.
1	WarmStart	The sending SNMP entity has reinitialized itself, but the agent's configuration has not been altered. This is typically a routine restart.
2	linkDown	The communications link has failed.
3	linkUp	The communications link has come up.
4	Authentication	The agent has received an incorrect community name. Failure from a manager.
5	EGP Neighbor	The external gateway protocol (EGP) neighbor is down. Loss.
6	Enterprise	An enterprise-specific event has occurred. Specific (Requires a specific trap type to identify).

## I want to receive and display SNMP traps from third party devices in the Event Console?

### Products:

- Unicenter TNG

Version: 2.1, 2.2, 2.4

OS: WIN/NT, WIN2000

### Solution:

To receive and display SNMP traps from third party devices in the Unicenter Event Console, you must enable the SNMP Trap Service. Select **Unicenter Enterprise Management > Configuration > Settings**. Change the **SNMP Trap Server Activated** flag to **Yes**.



# 9

---

## **Sending Mercury Business Availability Center Alerts to HP Openview VantagePoint Operations**

This chapter explains how to configure Mercury Business Availability Center to send alerts to HP Openview VantagePoint Operations.

<b>This chapter describes:</b>	<b>On page:</b>
About Sending Mercury Business Availability Center Alerts to HP Openview VantagePoint Operations	88
Verifying the Presence of the HP OVO Agent	88
Installing the HP OVO Agent on the Core Server Machine	89
Assigning and Distributing the opcmsg Template to the Mercury Business Availability Center Host	90
Sending a Mercury Business Availability Center Alert Using the HP OVO Agent opcmsg Command	91

## About Sending Mercury Business Availability Center Alerts to HP Openview VantagePoint Operations

To send alerts to HP Openview VantagePoint Operations (OVO), you must install an HP agent on the Core Server. Preconfigure the agent to send events to the management server via **opcmsg**. In HP terminology this means that the agent should have the **opcmsg** template assigned and distributed to it.

**To send alerts to HP OVO, verify the following:**

- ▶ An HP agent is installed on your Core Server host. For details, see the next section.

If no HP agent is installed, you must install it. For details, see “Installing the HP OVO Agent on the Core Server Machine” on page 89.

- ▶ The HP agent is configured to send events to the HP OVO management server via **opcmsg**. For details, see “Assigning and Distributing the opcmsg Template to the Mercury Business Availability Center Host” on page 90.

## Verifying the Presence of the HP OVO Agent

When sending alerts to the HP OpenView Management Server, Mercury Business Availability Center utilizes HP’s **opcmsg** command. This is a command that is used by HP OVO agents to communicate with the management server. You verify that an HP OVO agent is installed on the Mercury Business Availability Center server by running **opcmsg**.

**To verify the availability of the opcmsg command:**

- 1** On the Core Server, open a command window.
- 2** In the command window write **opcmsg** and press **Enter**. The possible results of this command are as follows:
  - ▶ **opcmsg** is not recognized as an internal or external command, operable program or batch file.

**Reason and Solution:** The HP OVO agent is not installed on the machine. Follow the steps in “Installing the HP OVO Agent on the Core Server Machine” on page 89 and “Assigning and Distributing the opcmsg Template to the Mercury Business Availability Center Host” on page 90.



- ▶ The following message is displayed:  
**The ITO Message Command is not configured on this system. Contact your ITO Administrator to configure it. (OpC30-913)**

**Reason and Solution:** The HP OVO is installed on the machine but is not configured to send **opcmsg** messages. Follow the steps in “Assigning and Distributing the opcmsg Template to the Mercury Business Availability Center Host” on page 90.

- ▶ The following message is displayed:  
**The application parameter is required. (OpC30-903)**  
**The object parameter is required. (OpC30-904)**  
**The msg\_text parameter is required. (OpC30-905)**

**Usage:**

```
opcmsg [-help] [-id] severity=normal|warning|minor|major|critical]
application=<application> object=<object> msg_text=<text>
[msg_grp=<message group>] [node=<node>] [service_id=<svcid>]
[-option <var>=<value>]* (OpC30-900)
```

**Reason:** The HP OVO is installed on the machine and is configured to send **opcmsg** messages.

## Installing the HP OVO Agent on the Core Server Machine

Make sure that the Core Server machine host is a part of a node group in HP OVO or you will not be able to see any of the alerts.

**To install the HP OVO Agent:**

- 1** On the Mercury Business Availability Center server, verify that an FTP service is running with write permissions.
- 2** On the HP OVO Management Server, locate the host in the HP OVO ITO Node Bank window and select it. If the host is not defined, you must first define it.
- 3** From the Actions menu, select **Agents** and then **Install/Update SW & Config**.

- 4 Select the **Agent Software** check box in the Components area. Verify that the host you selected appears in the list box on the right and that the **Nodes in list requiring update** check box above it is selected. Click **OK**.
- 5 In the window that opens, provide HP OVO with a user name and password that has administrative privileges for the Mercury Business Availability Center host.
- 6 Follow the on-screen instructions. You will need to point to the location of the file.
- 7 On the Mercury Business Availability Center server, note that you have to manually run the **opc\_inst.bat** file placed in a temporary directory in your FTP root directory.

## Assigning and Distributing the opcmmsg Template to the Mercury Business Availability Center Host

The following procedure explains how to assign and distribute the opcmmsg template to a Mercury Business Availability Center host. Once you have performed this procedure, you can set up Mercury Business Availability Center to send alerts to HP OVO. For details, see “Sending a Mercury Business Availability Center Alert Using the HP OVO Agent opcmmsg Command” on page 91.

### To assign and distribute opcmmsg:

- 1 On the HP OVO Management Server, locate the Core Server host in the HP OVO ITO Node Bank and select it. Select **Actions > Agents > Assign templates**.
- 2 Click **Add** and verify that the probe host appears in the list box to the left.
- 3 Click the **Open Template Window** button.
- 4 In the template window, select **Default**, double-click the appropriate operating system name, and select **opcmmsg(1|3)**.
- 5 Return to the Add Configuration window and click **Get template selection**. You should now see **opcmmsg(1|3)** appearing in the list box to the right.
- 6 Click the **OK** button in the Add Configuration and Define Configuration windows.

**7** Return to the Node Bank window and select **Agents > Install/Update SW & Config** from the Actions menu. Verify that the Mercury Business Availability Center host is still selected.

**8** Select the templates check box, and click **OK**.

A message should be displayed in your HP OVO message browser reporting that the templates have been distributed successfully to the host.

## **Sending a Mercury Business Availability Center Alert Using the HP OVO Agent `opcmsg` Command**

---

**Note to Mercury Managed Services customers:** This procedure is not relevant for Mercury Managed Services customers. Instead, Mercury Managed Services customers should use the SiteScope command line monitor. To use that monitor, consult Mercury Managed Services Support.

---

This command line utility is available after you install the HP OVO Agent on the Mercury Business Availability Center Core Server machine. For details, see “Installing the HP OVO Agent on the Core Server Machine” on page 89.

You use the `opcmsg` command to send a message to the Windows console or the system console without waiting for a reply.

**To configure Mercury Business Availability Center alerts:**

- 1** To configure the alerts that are to be sent to the HP OVO host, access the Alert Wizard: Select **Admin > Platform > Alerts and Recipients > Alerts**.
- 2** Create an alert according to the instructions in “Creating an Alert Scheme” in *Platform Administration*.

- 3 Continue to the section “Running an Executable File When the Alert is Triggered” in *Platform Administration*. During this procedure, you must select the **Run executable file** check box and define the alert action in the Alert Wizard. When asked to enter the command line required to run the executable file, use the HP OVO Agent `opcmsg` command line:

```
opcmsg [-help] [-id] severity=normal|warning|minor|major|critical]
application=<application> object=<object> msg_text=<text>
[msg_grp=<message group>] [node=<node>] [service_id=<svcid>]
[-option <var>=<value>]* (OpC30-900)
```

The `opcmsg` command defines three mandatory parameters: object, application, and message text. In addition you can use:

**severity.** Can be one of the following severities: normal, warning, minor, major, or critical.

**msg\_grp.** The name of the message group defined in HP OVO.

**node.** Used to define the node from which this alert was sent.

**service\_id.** Used to assign the alert to a specific HP Service Navigator service.

**-option <var>=<value>.** One or more key value pairs used by the OVO’s automatic and operator-initiated actions.

# 10

---

## Sending Mercury Business Availability Center Alerts to Tivoli TEC

This chapter explains how to configure Mercury Business Availability Center to send alerts to Tivoli TEC.

This chapter describes:	On page:
About Sending Mercury Business Availability Center Alerts to Tivoli TEC	93
Setting Up Tivoli TEC	94
Sending a Mercury Business Availability Center Alert Using the Tivoli End Point postmsg Command	95

### About Sending Mercury Business Availability Center Alerts to Tivoli TEC

You can integrate alerts with the IBM Tivoli Enterprise Console (TEC) application.

To send alerts to the Tivoli TEC management server, Mercury Business Availability Center utilizes Tivoli's **postzmsg**, **postmsg**, **wpostzmsg**, or **wpostmsg** commands. For details on these commands, refer to the Tivoli Web site ([http://publib.boulder.ibm.com/tividd/td/tec/SC32-1232-00/en\\_US/HTML/ecormst02.htm](http://publib.boulder.ibm.com/tividd/td/tec/SC32-1232-00/en_US/HTML/ecormst02.htm)). The recommended command to use is **postzmsg** as it can be copied to and run from any machine and it buffers events if the TEC server is unavailable.

The **wpostzmsg** and **wpostemsg** commands are commands used by the managed node to communicate with the management server. Therefore the Mercury Business Availability Center server must be a Tivoli End Point (that is, the Tivoli software agent must be installed on the machine).

The command used is called from the **run\_exe** command.

## Setting Up Tivoli TEC

To send alerts to Tivoli TEC, you must verify that:

- ▶ Tivoli End Point is installed on the Core Server host, if using the **wpostzmsg** or **wpostemsg** commands.
- ▶ An ACP profile has been distributed to the Tivoli End Point on the Core Server host.
- ▶ The command being used is located on the machine from which the alert is being sent.
  - ▶ If using the **postzmsg** or **postemsg** command, it can be located anywhere on the machine. You can verify that the command is present by running the following:

```
Usage: postzmsg { -S <server> | -f <config_file> } [-r <severity>]
[-m <message> ] [<slot_name=value>, ...] <class> <source>
```

- ▶ If using the **wpostzmsg** or **wpostemsg** command, it should be located under the directory `<Drive>:\Program Files\Tivoli\lcf\bin\<platform>\bin\`. Using one these commands requires the Tivoli environment to be sourced, so the command must be run from a batch file.

**To send an alert to Tivoli TEC:**

- 1 Create a **baroc** file which describes the event class of the alert, and deploy it on your Tivoli Tec. For more information on the baroc file format, refer to the Tivoli TEC documentation.

- 2 Copy the **postzmsg** or **postemsg** command to the Mercury Business Availability Center server. If using the **wpostzmsg** or **wpostemsg** command, install a Tivoli End Point on the Mercury Business Availability Center server (the End Point brings the **wpostzmsg** and **wpostemsg** executables with it).
- 3 Run the command by creating the appropriate command line. For example, if using the **postzmsg** command:

```
postzmsg -S server -r CRITICAL description="<AlertDescription>"  
triggerCause="<TriggerCause>" hostname="cookie"  
topazProfileName="<ProfileName>" Topaz_Alert Topaz
```

You can create and use a configuration file to specify various parameters. For details on running the commands, refer to the Tivoli Web site ([http://publib.boulder.ibm.com/tividd/td/tec/SC32-1232-00/en\\_US/HTML/ecormst02.htm](http://publib.boulder.ibm.com/tividd/td/tec/SC32-1232-00/en_US/HTML/ecormst02.htm)).

## Sending a Mercury Business Availability Center Alert Using the Tivoli End Point **postemsg** Command

---

**Note to Mercury Managed Services customers:** This procedure is not relevant for Mercury Managed Services customers. Instead, Mercury Managed Services customers should use the SiteScope command line monitor. To use that monitor, consult Mercury Managed Services Support.

---

This command line utility is available after you install Tivoli End Point on the Mercury Business Availability Center Core Server machine.

You use the **postemsg** command to send a message to the Windows console or the system console without waiting for a reply.

**To configure Mercury Business Availability Center:**

- 1 To configure the alerts that are to be sent to the Tivoli TEC host, access the Alert Wizard: Select **Admin > Platform > Alerts and Recipients > Alerts**.

- 2 Create an alert according to the instructions in “Creating an Alert Scheme” in *Platform Administration*.
- 3 Continue to the section “Running an Executable File When the Alert is Triggered” in *Platform Administration*. During this procedure, you must select the **Run executable file** check box and define the alert action in the Alert Wizard. When asked to enter the command line required to run the executable file, use the Tivoli End Point `postemsg` command line:

```
postemsg { -S <server> | -f <config_file> } [-r <severity>]  
[-m <message> ] [<slot_name=value>, ...] <class> <source>
```



---

# Index

## A

### alerts

- sending to BMC PATROL 75
- sending to CA Unicenter 79
- sending to HP OVO VantagePoint Operations 88

### API 7

- browser querying 10
- byTime function 23
- CMDB 39
- configuration 24
- creating queries 8
- data returned 10
- example queries 24
- maximum number of rows returned, setting 11
- metadata for samples 9
- OpenAPI Query Builder 13
- permissions 9
- queries, creating 19
- queries, legacy data 25
- query limitations 21
- SQL syntax supported 20
- time formats 22
- Web browser response body 11
- Web service, calling 9

## B

### BMC PATROL

- installation troubleshooting 76
- installing the SNMP adapter 76
- sending alerts to 75

## C

### CA Unicenter

- configuration troubleshooting 83
- sending alerts as SNMP trap 80
- sending alerts as SNMP traps with Unicenter cawto command 81
- sending alerts to 79

### CMDB API 39

- exceptions 42
- IDs 42
- parameter format 42
- permissions 41
- update methods 45, 47
- using 40
- Web service, calling 41

## E

### EMS adapters

- introducing 73

### Excel reports

- queries, creating 19

## F

### filter parameters 36

## G

### Generic Data Engine API 7

- using 8

## H

- HP OVO VantagePoint Operations
  - deploying opcmsg template 90
  - installing HP OVO agent on Core Server machine 89
  - sending alerts to 88
  - sending alerts to Mercury Business Availability Center 87
  - sending Mercury Business Availability Center alert using opcmsg command 91
  - verifying presence of HP OVO agent 88

## O

- OpenAPI 7
- OpenAPI Query Builder 13
  - accessing 13
  - introducing 13
  - permissions 14

## Q

- queries
  - creating using OpenAPI Query Builder 14
  - examples 37
  - for Generic Data Engine 19
  - limitations 21
  - structure 25

## R

- reporting
  - CMDB API 39
  - Generic Data Engine API 7
- rum\_page\_t sample
  - example 25
- rum\_server\_t sample
  - example 25

## S

- SNMP trap
  - enabling CA Unicenter 80

- ss\_t sample
  - example 24

## T

- third-party integrations
  - introducing 3
- Tivoli TEC
  - sending Mercury Business Availability Center alerts to 93
  - sending Mercury Business Availability Center alerts using postemsg command 95
  - setting up 94
- trans\_t sample
  - example 24

## W

- Web service
  - CMDB API 39
  - Generic Data Engine API 7
  - Generic Data Engine API, data returned 10
  - Generic Data Engine API, queries 19