# HP OpenView Business Process Insight

For the Windows® Operating System

Software Version: 02.10

## Reference Guide

Document Release Date: January 2007
Software Release Date: January 2007

# Legal Notices

## Warranty

The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

## Restricted Rights Legend

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

## Copyright Notices



## Trademark Notices

Java™ is a US trademark of Sun Microsystems, Inc.

Microsoft® is a US registered trademark of Microsoft Corporation.

Oracle ® is a registered US trademark of Oracle Corporation, Redwood City, California.

UNIX® is a registered trademark of The Open Group.

## Documentation Updates

This manual's title page contains the following identifying information:

- Software version number, which indicates the software version
- Document release date, which changes each time the document is updated
- Software release date, which indicates the release date of this version of the software

To check for recent updates, or to verify that you are using the most recent edition of a document, go to:

**http://ovweb.external.hp.com/lpe/doc_serv/**

You will also receive updated or new editions if you subscribe to the appropriate product support service. Contact your HP sales representative for details.

## Support

Please visit the HP OpenView support Web site at:

**http://www.hp.com/managementsoftware/support**

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

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

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

Most of the support areas require that you register as an HP Passport user and log in. Many also require a support contract.

To find more information about access levels, go to:

**http://www.hp.com/managementsoftware/access_level**

To register for an HP Passport ID, go to:

**http://www.managementsoftware.hp.com/passport-registration.html**

# Contents


1 Introduction . . . . . 9

2 OVBPI Architecture . . . . . 11

High-Level Component Architecture . . . . . 12

Business Process Dashboard . . . . . 13

The Dashboard and JSPs . . . . . 15

The Dashboard and Business Process Metrics . . . . . 15

Email Notifications and the Dashboard . . . . . 16

Intervention Client . . . . . 16

OVBPI Database . . . . . 18

Flow Data . . . . . 20

Business Process Metric Data . . . . . 20

Metric Threshold Definitions . . . . . 21

Event Hospital Data . . . . . 21

Model Repository Data . . . . . 21

Email Notification Data . . . . . 21

OVBPI Server . . . . . 22

Business Impact Engine . . . . . 24

Business Process Metric Definer . . . . . 28

Metric Engine . . . . . 30

Repository Explorer . . . . . 32

Notification Server . . . . . 33

Business Event Handler . . . . . 36

Modeler . . . . . 40

Repository Server . . . . . 41

Administration Consoles and Interfaces . . . . . 42

OVBPI Security . . . . . 43

Where to go Next . . . . . 43

3 Integrating with OVBPI . . . . . 45
OVBPI Integration Points . . . . . 46
OVBPI Adapters . . . . . 50
OVBPI and OVIS Probes and Alarms . . . . . 50
OVBPI OpenView Operations Adapter . . . . . 52
HP OpenView Service Desk Adapters . . . . . 53
OVBPI OpenView SOA Manager Adapter . . . . . 54
iWay Integration . . . . . 56
OpenView Dashboard . . . . . 57
Self-Healing Services . . . . . 59
4 OVBPI and OVIS . . . . . 61
OVBPI and OVIS Integration . . . . . 63
OVBPI and OVIS Design-Time Integration . . . . . 63
OVBPI and OVIS Run-Time Integration . . . . . 65
Reporting on the Operational Status of Your OVBPI System . . . . . 68
Reporting SLO and SLA Violations . . . . . 68
Custom Probes . . . . . 69
Creating a New Service Group for an OVBPI Service . . . . . 71
Configuring Alarms and SLOs . . . . . 79
Defining Objective Information for OVBPI Monitored Services . . . . . 79
Defining Service Level Agreements for OVBPI Monitored Services . . . . . 85
Making Sure OVIS Adds Alarm Data to Its Database Tables . . . . . 86
Defining Probe Locations . . . . . 86
Configuring Probes for Multiple OVBPI Servers . . . . . 87
5 OVBPI and OVSD . . . . . 89
OVBPI and OVSD Service Call and Incident Information . . . . . 90
OVSD Web API . . . . . 91
Automatic OVSD Service Mapping . . . . . 92
Defining OVSD Custom Fields . . . . . 92
6 OVBPI and SOA Manager . . . . . 97
SOA Manager and OVBPI Integration . . . . . 98
Business Events and SOA Manager . . . . . 101
Configuring Access to the SOA Manager Adapter . . . . . 102

A Database Schemas . . . . . 105
Building Applications to Use the OVBPI Metrics Data with Microsoft SQL Server . . . 106
Oracle and SQL Server Data Type Definitions . . . . . 107
Business Metrics Schema . . . . . 108
Alerts Facts . . . . . 108
Facts Values . . . . . 112
Statistics Facts . . . . . 116
Dimension Tables . . . . . 121
Business Metric Custom Types . . . . . 134
Business Metric Failure Messages . . . . . 135
Metric Views . . . . . 135
Metric Values . . . . . 137
Flow Schema . . . . . 139
Flows . . . . . 140
Flow Instance . . . . . 142
Nodes . . . . . 146
Node Instance . . . . . 148
Node Instance Started Times . . . . . 150
Node Instance Completed Times . . . . . 151
Arcs . . . . . 152
Services . . . . . 152
Node2Resources . . . . . 154
Business Entity Schema . . . . . 157
Data item names . . . . . 157
Data types . . . . . 157
Keys . . . . . 157
Data Objects . . . . . 158
Data Definition Instances . . . . . 159
Business Event Handler Schemas . . . . . 161
Business Event Handler Event Store . . . . . 161
Event Hospitals . . . . . 163
Complete List of OVBPI Database Tables . . . . . 166
B Expression Grammar in Flow, Data and Filter Definitions . . . . . 171
Grammar . . . . . 172

Function Return Values . . . . . 174
Functions for Dates and Times . . . . . 174
Functions for Identifying String Values . . . . . 174
Functions for All Property Types . . . . . 175
Flow Progression Rules . . . . . 176
Methods for Progression Rules . . . . . 178
Case Sensitivity for Expressions . . . . . 182
Expression Properties . . . . . 182
Expressions with String Constants . . . . . 182
C Coercion Rules . . . . . 183
Assignments . . . . . 184
Expressions . . . . . 185

# 1 Introduction

This guide provides reference information relating to OpenView Business Process Insight (OVBPI). This reference information covers OVBPI and the components that OVBPI integrates with as follows:

- Chapter 2, OVBPI Architecture

  This chapter describes the architecture of the OVBPI system and how the OVBPI components relate to each other.

- Chapter 3, Integrating with OVBPI

  This chapter describes the different integration points within your OVBPI system.

- Chapter 4, OVBPI and OVIS

  This chapter provides details of how OVBPI integrates with HP OpenView Internet Services (OVIS), specifically, to configure customized OVBPI probes for OVIS.

  Installing OVIS probes is an optional task, and if you want to use OVIS to report OVIS metric information relating to OVBPI flows, you need to install the probes and configure them as described in Chapter 4, OVBPI and OVIS.

  You can also configure an OVBPI Server to receive service impact information from OVIS. You configure your OVBPI Server to receive these impact reports using the Administration Console as described in the *OVBPI System Administration Guide*.

- Chapter 5, OVBPI and OVSD

  If HP OpenView Service Desk (OVSD) is a component in your system, you can configure OVBPI to link to OVSD Incident reports and Service Calls, and display their details through the Business Process Dashboard. This chapter provides details of how OVBPI integrates with OVSD through the OVBPI Dashboard in order that you can achieve this.

- Chapter 6, OVBPI and SOA Manager

  This chapter provides details of how OVBPI integrates with HP OpenView SOA Manager. It describes how to import the required SOA Manager business services and link them to your business flows.

  You also configure your OVBPI system to receive SOA business events by configuring an event source for SOA Manager. This is described in the *HP OpenView Business Process Insight Training Guide - Business Events*.

- Appendix A, Database Schemas

  This appendix details the database schemas that are defined to generate reports from the OVBPI impact data and for use by the Business Impact Engine, Metric Engine and Business Events Handler.

- Appendix B, Expression Grammar in Flow, Data and Filter Definitions

  This appendix lists the rules for the grammar that can be used for business flow progression rules and expressions, and expressions within business process metric filter definitions.

- Appendix C, Coercion Rules

  This appendix describes the rules for how properties are coerced when evaluating binding, filter and assignment expressions within the OVBPI Modeler.

# 2 OVBPI Architecture

This chapter provides a high-level description of the architecture of the OVBPI system. The architecture is presented to provide an understanding of the components and concepts that are described in later sections and chapters. You can read the early sections of this chapter to gain an overview of the architecture, and then reread this chapter later, if you want more detail.

This chapter also lists which components can be customized and where to find more information about them.

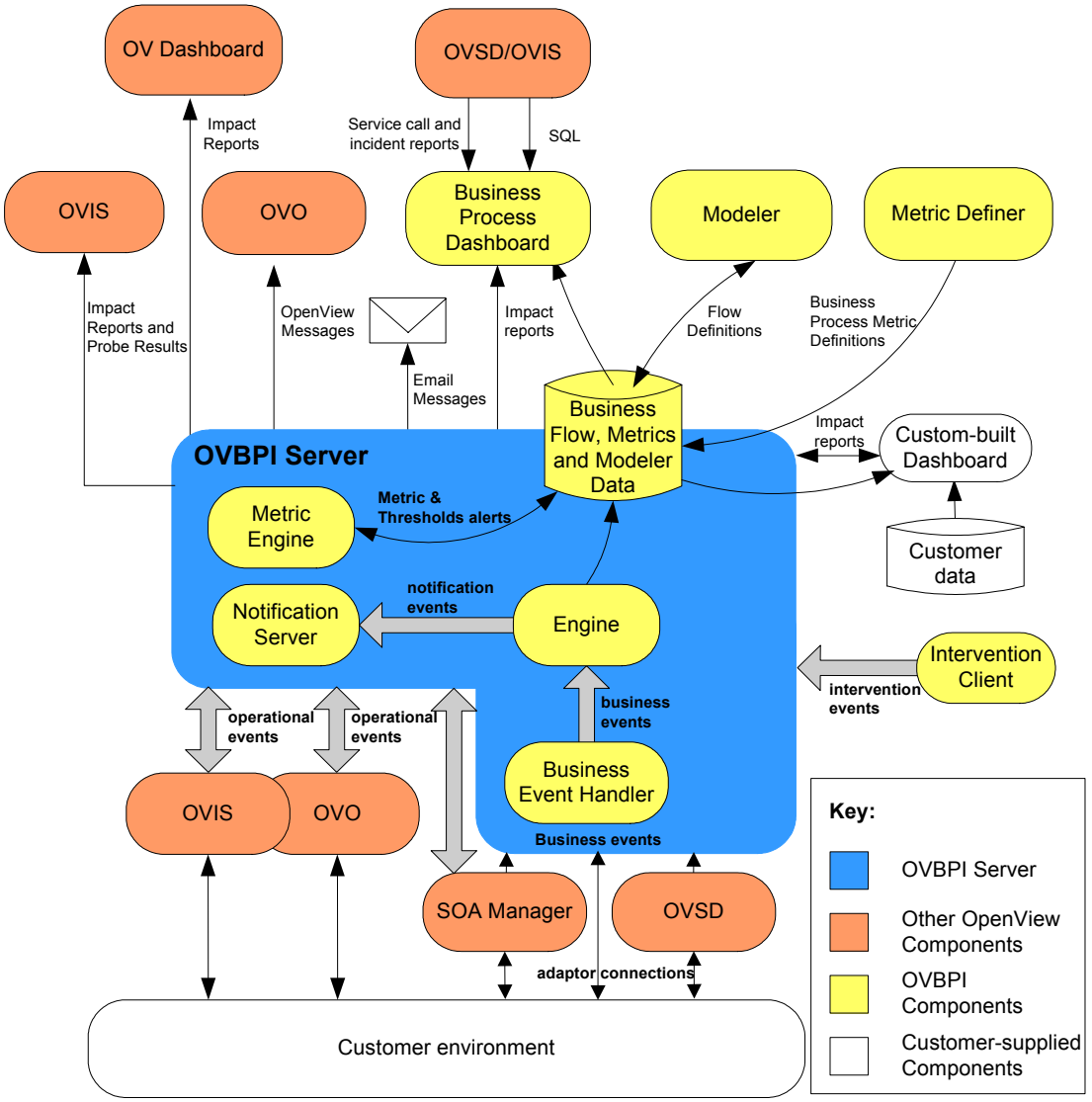
Specifically, the chapter covers the following topics:

- A high-level representation of the OVBPI component architecture; see Figure 1 on page 12.
- The OVBPI Business Process Dashboard, which is used to display the impact information from the OVBPI system; see section Business Process Dashboard on page 13.
- The OVBPI Intervention Client, which provides you with access to active business flows and flow data; see section Intervention Client on page 16.
- The OVBPI database and how it is used by the OVBPI components; see section OVBPI Database on page 18
- The Metric Definer, which you use to define business process metrics and metric thresholds for your flows; see section Business Process Metric Data on page 20 and section Metric Threshold Definitions on page 21.
- The OVBPI Server components; see section OVBPI Server on page 22.
- The OVBPI Modeler and the data used and stored by the OVBPI Modeler; see section Modeler on page 40.
- The administration GUIs and consoles used within the OVBPI system; see section Administration Consoles and Interfaces on page 42.

# High-Level Component Architecture

Figure 1 shows a high-level diagram of the OVBPI architecture.

**Figure 1 OVBPI Architecture**



# Business Process Dashboard

The OVBPI Business Process Dashboard is a Web-based interface for viewing the progress of your business flows. It enables you to monitor business flows and flow instances, including the business events, operational services and business process metrics that you have modeled and deployed using the OVBPI Modeler and the Metric definer.

The OVBPI Dashboard also enables you to associate the service information received from OpenView Operations (OVO), OpenView Internet Services (OVIS) and OpenView SOA Manager with OpenView Service Desk (OVSD) Service Calls and Incidents; see Chapter 5, OVBPI and OVSD.

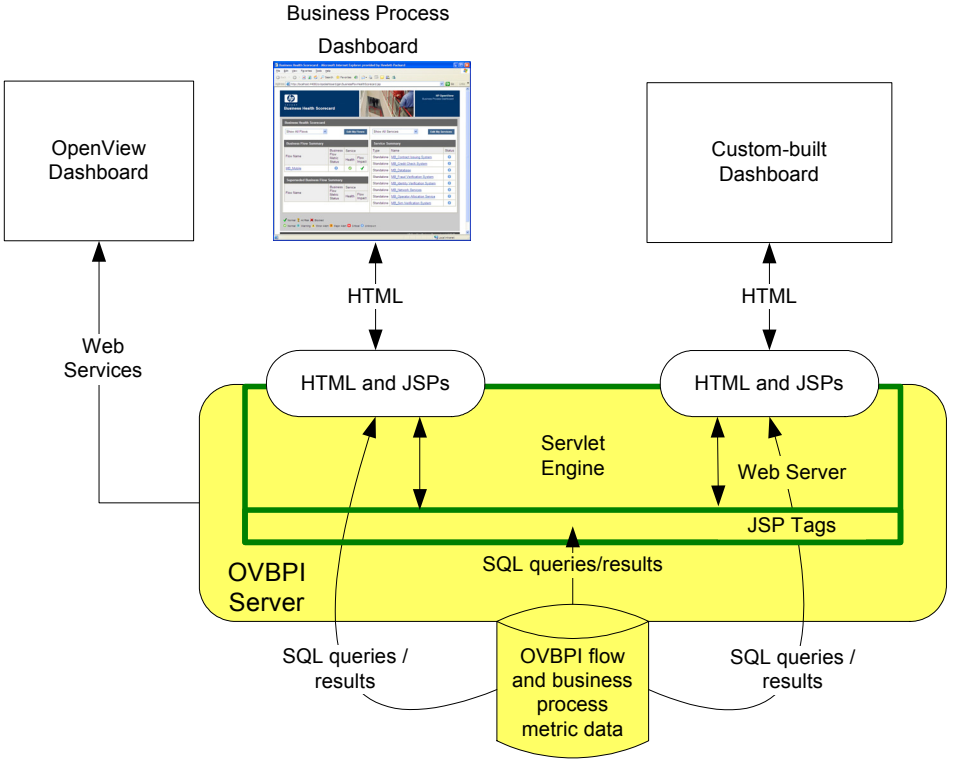
Specifically, the OVBPI Dashboard provides the following through a number of different views and graphical displays:

- An overall business health scorecard, which provides an overall view of the health of business flows that you have deployed.
- Flow instance information related to the flows that you have deployed.
- Business process metric information and threshold alerts related to individual flows and flow instances that are deployed.
- Reports based on the statistical information collected from the business process metrics that you have defined.
- Information relating to the status of the operational Services that your flows are linked to within OVO, OVSD, SOA Manager and OVIS.
- HP OpenView Service Desk Service Calls and Incidents for operational Services, where the information is available.
- Where appropriate, details of the status of your 60-day `Instant On` license. The Dashboard displays the number of days left before the `Instant On` license expires and OVBPI stops running.

If you have installed the OVBPI Business Process Dashboard on a different system to the OVBPI Server, you do not get information about the status of your license.

The OVBPI Dashboard comprises a set of HTML pages, which are created using Java Server Pages (JSP). JSP is a simple way to create dynamic Web pages, which are both platform independent and server independent. A Servlet Engine is used to manage these pages; this Servlet Engine is Tomcat and is installed with the OVBPI Server.

**Figure 2 OVBPI Business Process Dashboard**




The design of the OVBPI Dashboard means that it is possible to customize it for your specific business flows (see section The Dashboard and JSPs on page 15). This enables you to present data on your business flows within an interface that is tailored to your business area. However, you do not need to make any changes or customizations to the Dashboard as it can be used, without modification, to show information about any flows that you create.

Refer to the *OpenView Business Process Insight Integration Training Guide - Customizing the Business Process Dashboard* for information on how to customize the Dashboard and also how to create annotations for your business flows.

## The Dashboard and JSPs

The Business Process Dashboard uses a JSP custom tag library to communicate with the OVBPI components. This reduces to a minimum the amount of programming required to make modifications to the Dashboard. Refer to the *OpenView Business Process Insight Integration Training Guide - Customizing the Business Process Dashboard* for details of the JSP Tags and how to develop a personalized Dashboard based on these tags.

JSP technology separates content generation from presentation and takes advantage of reusable tags and objects, simplifying the maintenance of your Web applications. Using JSP enables you to access the OVBPI flow data and present it in a form of your choice to your business managers. This allows you to create queries and enhance the flow data with existing business data. This, combined with the JSP custom tag libraries, provides an easy way to modify the example Dashboard or develop a new dashboard to provide the impact information for a specific flow.

## The Dashboard and Business Process Metrics

The Business Process Dashboard displays tables, graphs and dials relating to the:

- overall health of the business flows relative to the business process metrics that are deployed within OVBPI.
- business process metrics and metric thresholds that you have configured, including the alert status and individual alerts.
- details of the individual flow instances and metric instances within the OVBPI system.
- historical statistical information relating to the business process metrics.

This business process metric data is displayed automatically through the OVBPI Dashboard as a result of your defining the business process metrics and business thresholds for your flows.

Business process metrics and metric threshold definitions are set up using the Metrics definer, which is described in more detail in the *HP OpenView Business Process Insight Integration Training Guide - Defining Business Process Metrics*.

## Email Notifications and the Dashboard

In addition to proactively monitoring business flows through the Dashboard, you can configure OVBPI to send email notifications relating to your flows through the Notification Server (see section Notification Server on page 33). These email notifications contain information relating to the business flows that you are monitoring.

The notification emails can also be configured to contain links to your OVBPI Dashboard, where you can then find out more about the status of your IT services and their impact on your business flows.

# Intervention Client

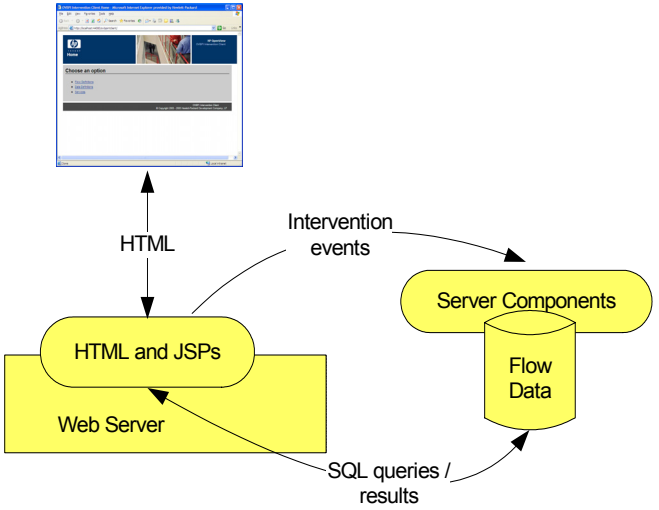
The Intervention Client enables you to access flows that you have deployed in order to modify or delete Flow instances and their associated Data instances. You might need to do this to resolve problems with the flow, or its data, usually after a period of new development or where progression rules are not behaving as expected. For example, you might need to:

- manually progress and delete flow instances.
- update and delete data instances.
- update the status that OVBPI records for an operational service.

  Note that the Intervention Client does not have an effect on the service as it is defined within OVO or OVIS; it makes changes only to the recorded status within OVBPI.

The Intervention Client is a secure, Web-based, console that uses Web authentication to enable you to make the modifications to your business flows.

**Figure 3 Intervention Client**




The Intervention Client is similar in architecture to the Business Process Dashboard in that it is a set of JSPs that use SQL to access the OVBPI database; see Figure 3 on page 17. The Intervention Client also uses Tomcat as its Servlet Engine and Web Server.

The Intervention Client is aimed at the person who is managing the OVBPI system and who has authority to make these modifications to the business flows. More details of the Intervention Client and how to use it in your solution are provided in the *OVBPI System Administration Guide*.

# OVBPI Database

OVBPI uses a relational database to record the following information:

- Flows and any related data
- Business process metrics
- Metric threshold definitions
- Metric threshold alerts
- Metric statistics
- Event hospital data, including the business event data
- Operational service status data
- Model Repository data
- Email notification data

This information is used by the OVBPI components for monitoring and progressing flow instances, plus reporting flow impact data, threshold violations and business process metrics through the Dashboard. It is also used by the Business Impact Engine to identify the Flow, Data and Service definitions that have been deployed using the OVBPI Modeler.

The database holds all the information that OVBPI needs to operate, with the exception of a number of configuration files, which are located under the OVBPI installation directory.

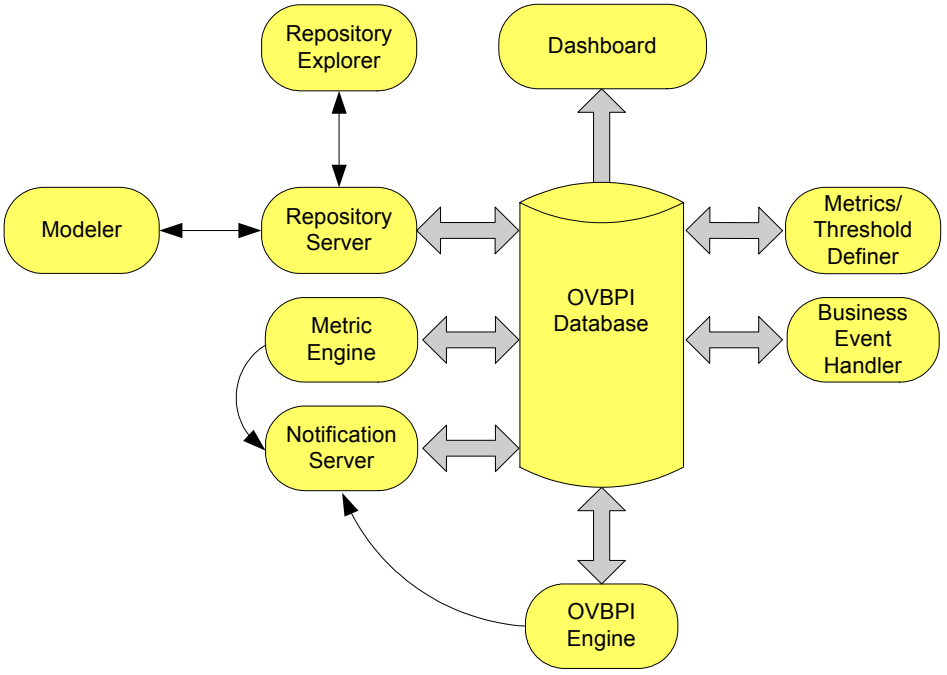
You can access the flow and business process metric data held in the OVBPI database directly, using SQL, and generate your own custom-built reports. You might want to do this in order to integrate these reports into your own reporting applications.

The data schema description that represents OVBPI information in the database is fully described in the *OpenView Business Process Insight System Administration Guide*. Be aware that this appendix describes only those tables that are supported for access. There are other tables, which are specifically for use by the OVBPI components and must not be modified, or accessed. These database tables are listed for completeness in the appendix, but are not described. As an example, you might need to know the name of all the tables that are relevant to OVBPI for backup and recovery purposes; however, the tables that are listed and not described must not be modified or used for reporting purposes.

Figure 4 shows a high-level diagram of the relationship between the database and the OVBPI components. The database can be installed and configured on the same system as an OVBPI Server, or on a system that is remote from the OVBPI Server; this is not shown in Figure 4.

Refer to the *OpenView Business Process Insight Installation Guide* for details of installing the schema or database files into a local or remote database.

**Figure 4 OVBPI Database**




The OVBPI components' use of the database tables is described in the following sections.

## Flow Data

This is the schema for the Business Impact Engine data. The OVBPI database maintains the state of the Engine database objects (or definitions), for example, the business Data definitions and business Flow definitions, within the OVBPI system. These definitions comprise status information required by the Business Impact Engine for:

- Flows
- Data
- Services

These Flow, Data and Service components, plus the Event information, are all defined through the OVBPI Modeler. The data required to populate the Event definitions is obtained from your business applications through the Business Event Handler.

Defining business flows is described in more detail in the *HP OpenView Business Process Insight Integration Training Guide - Modeling Flows*.

## Business Process Metric Data

The OVBPI database holds the data collected through the metric tables as a result of the progress of business flows and the status of the flows relative to these defined metrics. Business process metrics are evaluated as part of the flow data as nodes are progressed by the Business Impact Engine. A business process metric database table is populated using database triggers from the Nodes database table. The Metric Engine then processes the data from this metrics table and uses the results to calculate statistics and populate the remaining metrics tables. The results of these calculations are then presented to you through the Business Process Dashboard.

You can also use reporting applications to access the metric and statistical information in the database tables and generate reports from the data that is collected, or you can develop your own custom-built dashboard.

You can read more about the Metric definer and the metric data in the *OpenView Business Process Insight Integration Training Guide - Defining Business Process Metrics*. Building a customized Dashboard is described in the *OpenView Business Process Insight Integration Guide - Customizing the Business Process Dashboard*.

## Metric Threshold Definitions

You can optionally define thresholds for your business process metrics. If you do this you might also want to be notified when these thresholds fall outside acceptable values. OVBPI enables you to create metric threshold definitions for each business process metric so you can then be notified when these metric threshold values are violated.

Metric threshold alerts are shown:

- using the Business Process Dashboard
- as email alerts using your email system
- as OpenView Operations alerts from other OpenView applications

You can read more about metric thresholds and creating them using the Metric definer in the *OpenView Business Process Insight Integration Training Guide - Defining Business Metrics*.

## Event Hospital Data

Details of the events that are rejected by the Business Impact Engine because it does not recognize them are written to the Event Hospital table in the OVBPI database. From here they can be accessed, modified and resubmitted to the Business Impact Engine at a later date, if required.

## Model Repository Data

The design-time Flow modeling information is held in the Model Repository tables in the OVBPI database. This is the information that you enter using the OVBPI Modeler.

## Email Notification Data

These are the database tables relating to the information about the Notification Server subscriptions. The information in these tables is defined through the Notification Server Administration Console.

In addition to data relating to subscriptions, these tables contain data for the Notification Server retry mechanism.

# OVBPI Server

The OVBPI Server is the core of the OVBPI system; it is where the business and operational events are received, and their impact on the flows evaluated. The OVBPI Server is responsible for:

- Maintaining the Flow, Data, Event and business process metric definitions through the OVBPI database.
- Enabling management for the Model Repository data, including the ability to browse and print the content of the Repository.
- Monitoring business events, through adapters, in order to maintain the flow context for the business.
- Monitoring services and reporting any change that causes an impact on the flows.
- Defining business process metrics and metric thresholds.
- Monitoring metric thresholds and reporting on the threshold violations.
- Validating the status of flows according to progression rules that you have specified.
- Sending email notifications for flow impact alerts that have been subscribed to.
- Sending email notifications for metric threshold alerts that have been subscribed to.

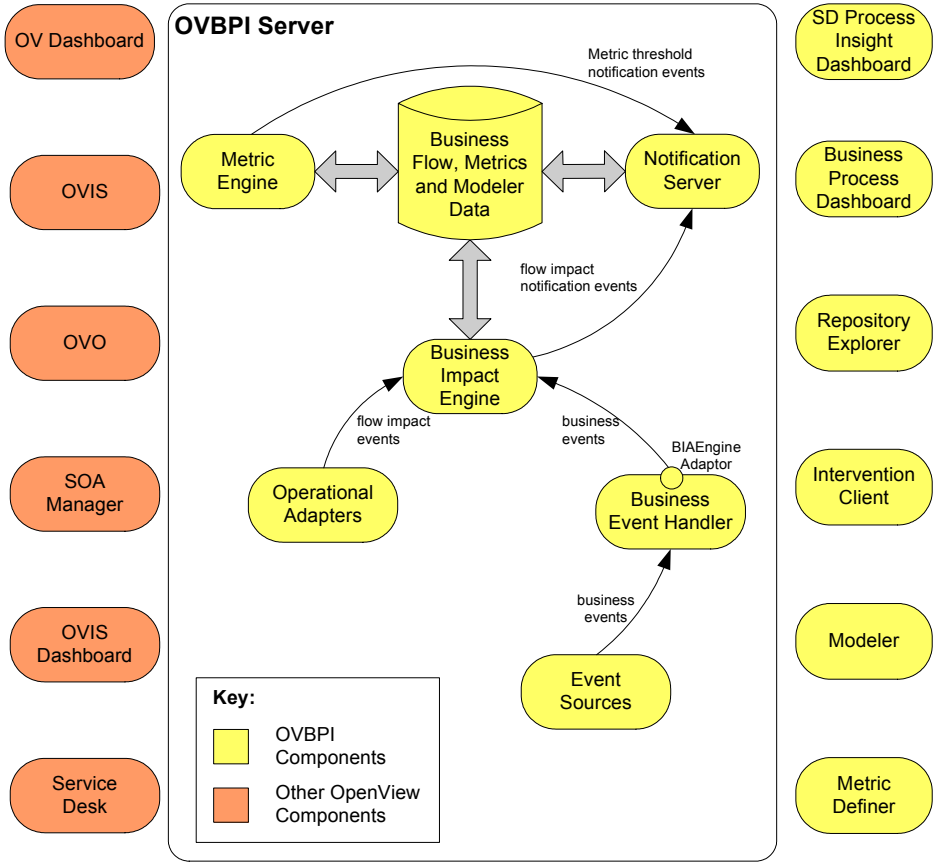
The components of the OVBPI Server that provide these functions are the:

- Business Impact Engine
- Metric Engine
- Business Process Metrics definer
- Repository Explorer
- Notification Server
- Business Event Handler

OVBPI can also accept service impact reports from other HP OpenView products; see section Chapter 3, Integrating with OVBPI.

The individual OVBPI Server components are shown in the following diagram and described, in more detail, in the following sections.

**Figure 5 OVBPI Server Architecture**

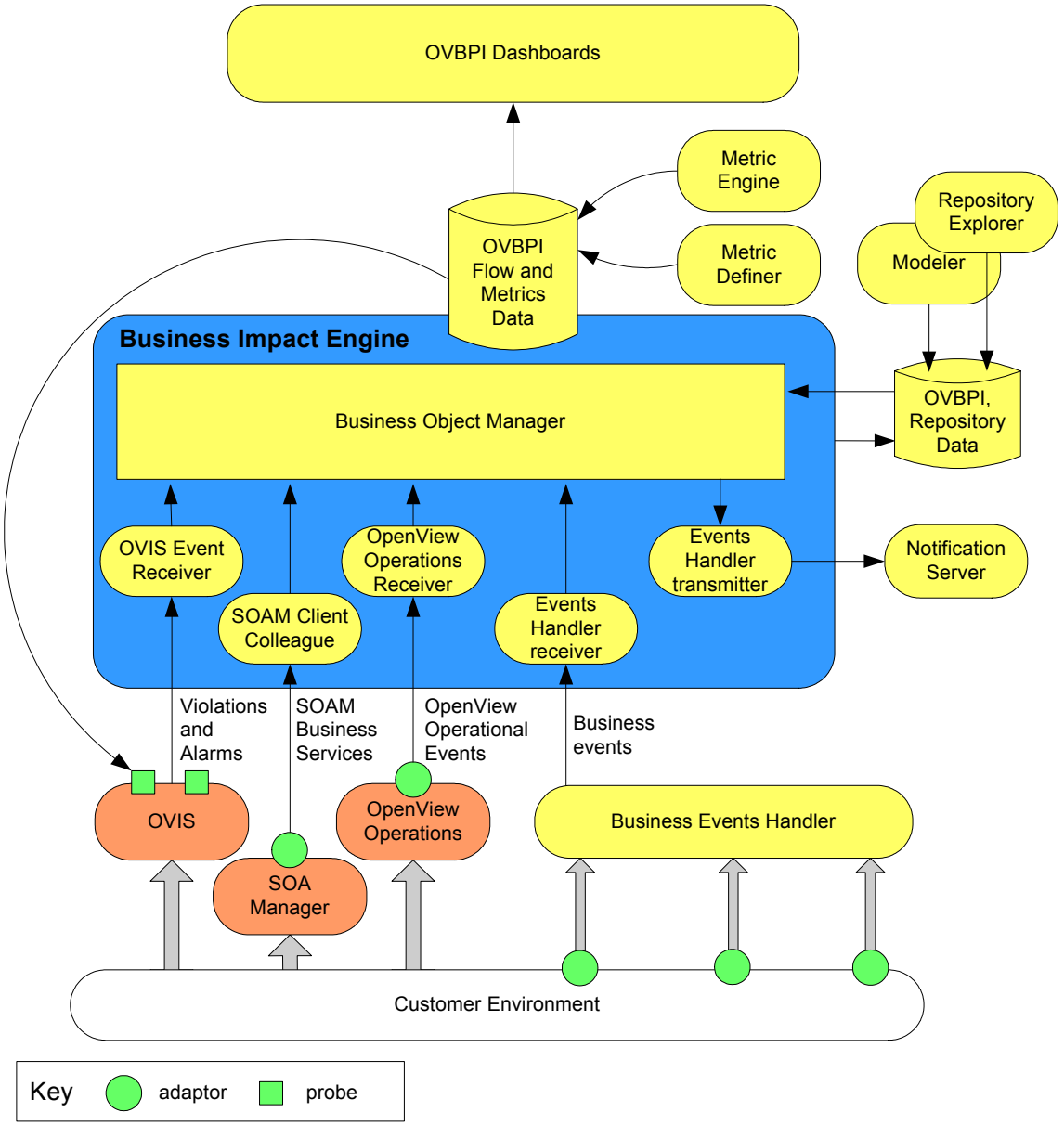


## Business Impact Engine

The purpose of the Business Impact Engine is to process operational and business events and derive business impact information from these events. Using the information defined in business flow definitions, the Business Impact Engine raises flow impact alerts when an underlying operational service or business event, which is relied on by the flow, falls outside defined thresholds.

Business Metric Threshold violations are managed by the Metric Engine as described in section Metric Engine on page 30.

**Figure 6 Business Impact Engine Architecture**



When a flow impact threshold violation is identified, the Business Impact Engine sends an impact alert to the Notification Server. In addition, the Business Process Dashboard checks the database for status changes resulting from these impact alerts to present the alerts to the business manager. This enables you either to be notified, or to proactively monitor the status of your business flows, according to your preferences and requirements.

The Business Impact Engine comprises the following components, which perform functions internal to OVBPI. These components are not generally exposed in any of the interfaces; however, you might see them referenced in log and error messages:

- Business object manager

  This manages the business objects within the Business Impact Engine, for example, the OVBPI business Data definitions and instances and the business Flow definitions and instances. The business object manager also receives events from OpenView Internet Services (OVIS), OpenView Operations and the Business Event Handler, plus specific internal events, such as flow progression events. Where appropriate, these events are propagated to the relevant business objects.

  The business object manager is generally a passive component and takes actions only when it receives an event. The exceptions are the Model Cleaner and the Engine Instance Cleaner, which are active.

- OVBPI OpenView Operations Receiver and Adapter

  The OVBPI OpenView Operations Receiver accepts operational status events from the OpenView Operations Adapter, converts them into an OVBPI event format and then passes them on to the business object manager.

  Operational events are status events, providing information on the status of the underlying applications, and system infrastructure, for example the CRM system, order processing system, routers or firewalls.

  The OVBPI OpenView Operations Adapter communicates with OpenView Operations to obtain the status of OpenView Operations (OVO) services. These are the OVO services that you link into your business flow using the OVBPI Modeler.

- SOA Manager Client Colleague

  The client colleague enables OVBPI to import SOA Manager business service definitions using the SOA Manager adapter. It then propagates the status events from these definitions in order to report on the status of the SOA Manager business services that you have configured.

- OVIS Event Receiver

  The OVIS Event Receiver polls for OVIS alarms and service level violations. These provide:

  — operational service status information in the form of alarms.

  — SLO and SLA violation information.

  As a result of receiving these alarms and violations, OVBPI can send email notifications to users configured to receive them through the Notification Server; these impact messages can have links to the relevant page in the Dashboard.

  The OVIS Event Receiver converts service impact alarms into OVBPI operational events, which then update the status of the appropriate service definition in the business object manager.

  The OVIS Event Receiver also converts SLA and SLO violations into OVBPI events and sends them directly to the Notification Server.

  OVIS alarms inform you of the status of your Internet and other services. This is in terms of how effectively they are running against the criteria that you configure within OVIS.

- Event Receivers and Transmitters

  The Event Receiver accepts business events, using RMI, as Java objects from the Business Event Handler, and passes them on to the Business Object Manager. There can be more than one Event Receiver, to improve performance; however there is only one Business Object Manager, which means there is a limit to the benefit of adding more Event Receivers.

  The Event Transmitter accepts alerts from the Business Object Manager, converts them into an RMI message and sends them to the Notification Server.

## Business Process Metric Definer

The Metric definer is a Web-based GUI that enables you to define business process metrics and metric threshold definitions for deployed flows.

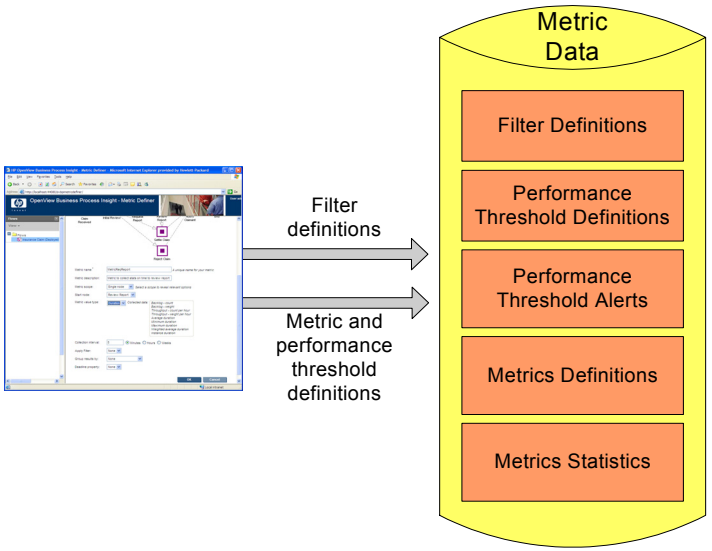
A business process metric is a business measurement that has a specific meaning within your business and can be used to record information about the business flow that you are monitoring. Statistics, such as average and maximum, can be calculated using metric instance data. In addition, metric thresholds can be set on both the individual metric instances and on the statistics generated from the metric instances. When metric thresholds are violated, threshold alerts are generated and can be reported through the Business Process Dashboard. You can also configure the Notification Server to send alert messages when a metric threshold is violated.

Within OVBPI, the process of defining business flows is separated from the process of defining business process metrics for these flows. You use the OVBPI Modeler to define your business flows and you use the Metric definer to define the business process metrics and metric thresholds for these business flows. You define business process metrics for a business flow after the flow has been deployed; this provides more flexibility for defining and modifying business process metrics without the need to redeploy your business flows.

You can also use filters to further target the statistical data that you want to collect. This has the added benefit of reducing the amount of data stored in the database and therefore maintaining the OVBPI system performance.

Figure 7 shows the Business Process Metric definer relative to the Business Metric data stored in the OVBPI database.

**Figure 7 OVBPI Business Process Metric Definer**




You can define metrics for:

- Single nodes
- Multiple nodes
- The whole flow

These metrics can be duration based or weight based. In addition, you can define a custom metric.

Data are collected based on this configuration, and thresholds can be defined to report on individual flow instances or on statistics for multiple instances, such as averages and standard deviations over a specified time period.

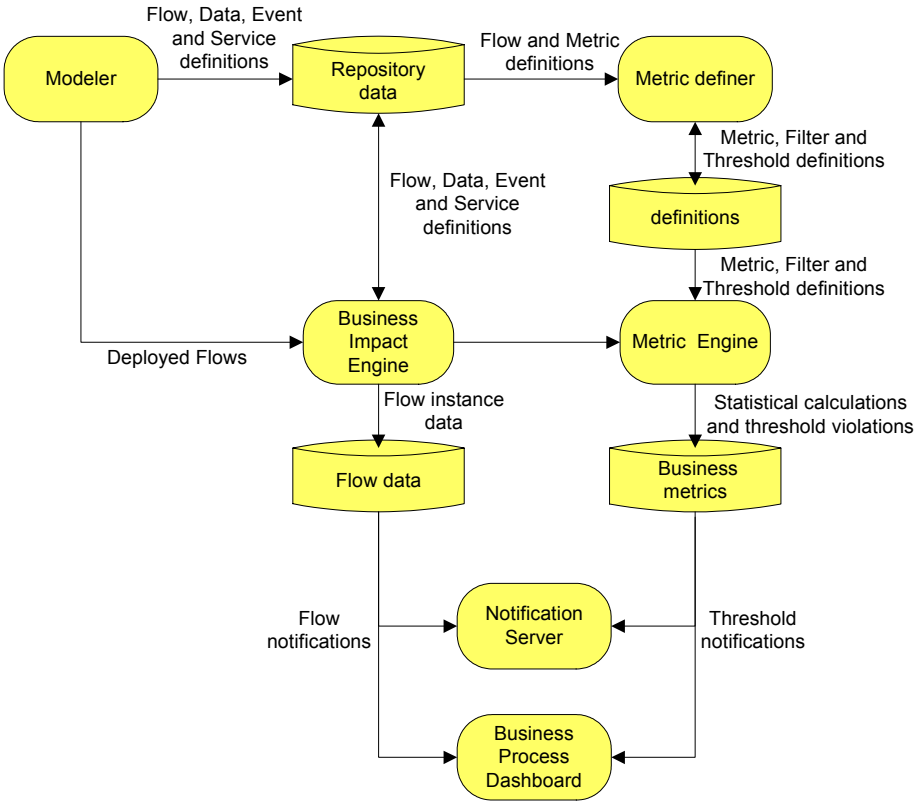
How the Business Process Metrics definer interworks with the Metric Engine is described in section Metric Engine on page 30.

## Metric Engine

The Metric Engine is the component of the OVBPI Server that analyzes and provides the statistical results from the business process metric and metric threshold data that you define. The Metric Engine takes data about the individual instances from the Business Impact Engine and combines this data with the data you enter through the Metric definer to collect and report on the required information about the flow.

Figure 8 shows the relationship between the flow impact data and the metric data that you define.

**Figure 8 Flow Impact and Metric Data Relationship**



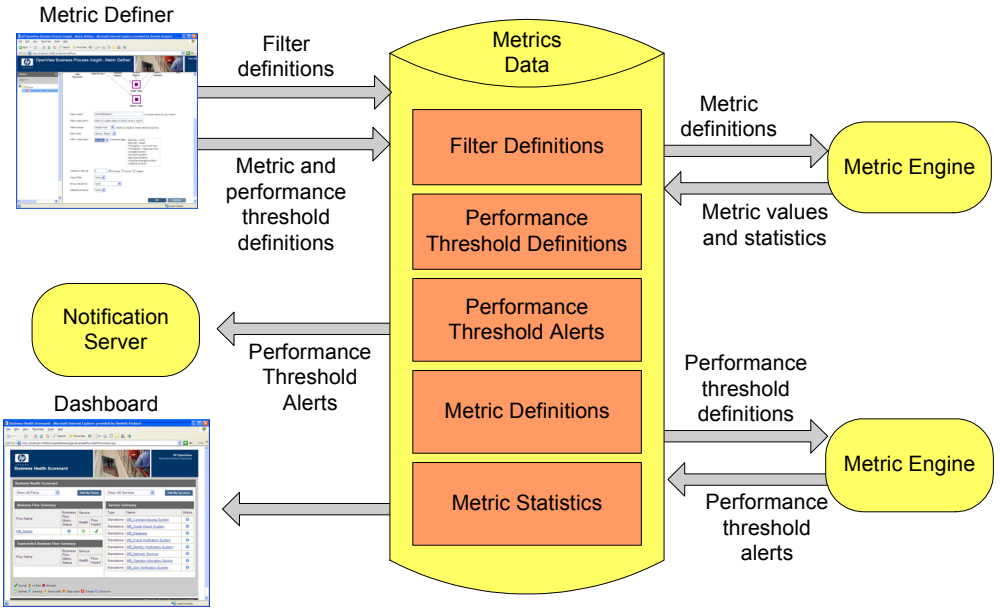

The business process metrics and metric thresholds are configured through the Business Process Metrics definer; see section Business Process Metric Definer on page 28.

The Metric Engine maintains business data for the business flows that are deployed according to the business process metrics that you have defined. It also maintains statistical information about the metric thresholds that are defined.

As an example, if you have defined a business process metric and metric threshold to record the backlog of flow instances at a specific node in the flow, the Metric Engine records and maintains the statistics required to calculate this backlog. It then reports the results through the OVBPI Dashboard.

Figure 9 shows how the metric data is used by the Metric Engine to report statistics and threshold violations to other OVBPI components.

**Figure 9 OVBPI Business Process Metric Engine**




The statistics that the Metric Engine calculates are stored in Metric database tables and are used by the OVBPI Dashboard and can also be used by other reporting tools if required. The database tables also hold information used to determine if any metric thresholds have been violated. These violations are reported through the OVBPI Dashboard and can also be reported through the Notification Server or other OVO applications.

## Repository Explorer

The Repository Explorer is a Web-based interface that enables you to view, print and manipulate the contents of the Model Repository. The Model Repository is a set of database tables that hold the data for the business processes that you have defined using the OVBPI Modeler.

Note that the data presented by the Repository Explorer might differ from the data that is deployed to the Business Impact Engine. This is because you might have modified the flow within the OVBPI Modeler, but not yet redeployed the flow.

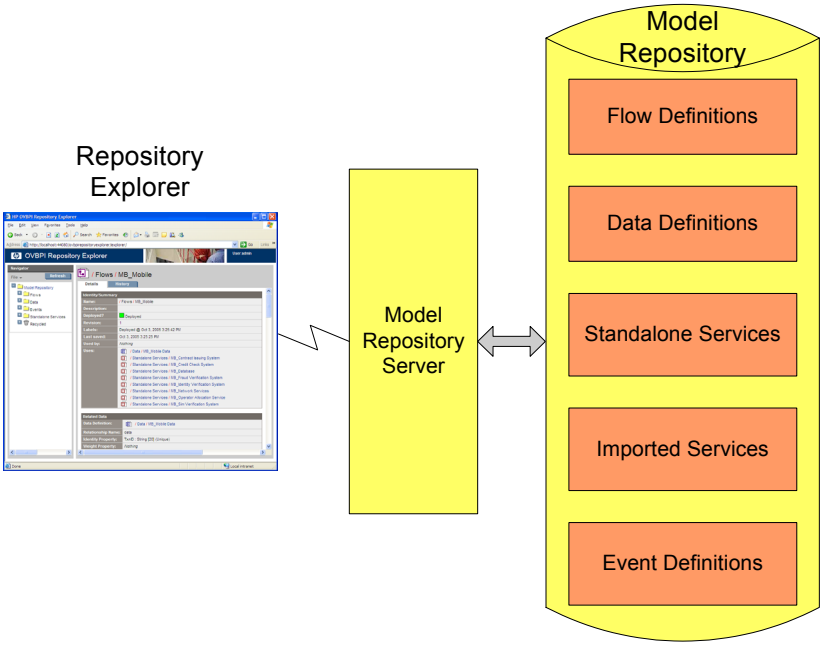
The Repository Explorer also provides access to information about all the superseded versions of the business flow that are currently deployed, enabling you to view and export details of the superseded flows.

Using the Repository Explorer you can:

- View all the details of your currently deployed and superseded flows in a tabular form.
- Print currently deployed and superseded definitions.
- Export the latest versions of your currently deployed definitions.
- Export the superseded versions of definitions.
- Remove (delete) superseded definitions.
- Import previously exported definitions.
- View, definitions that have been deleted using the OVBPI Modeler. These definitions appear in the Recycled folder within the Repository Explorer, from where they can be permanently deleted.

In summary, the Repository Explorer is a tool that you can use to browse and manage the Model Repository data and is accessed through the Repository Server as shown in Figure 10.

**Figure 10 Repository Explorer**



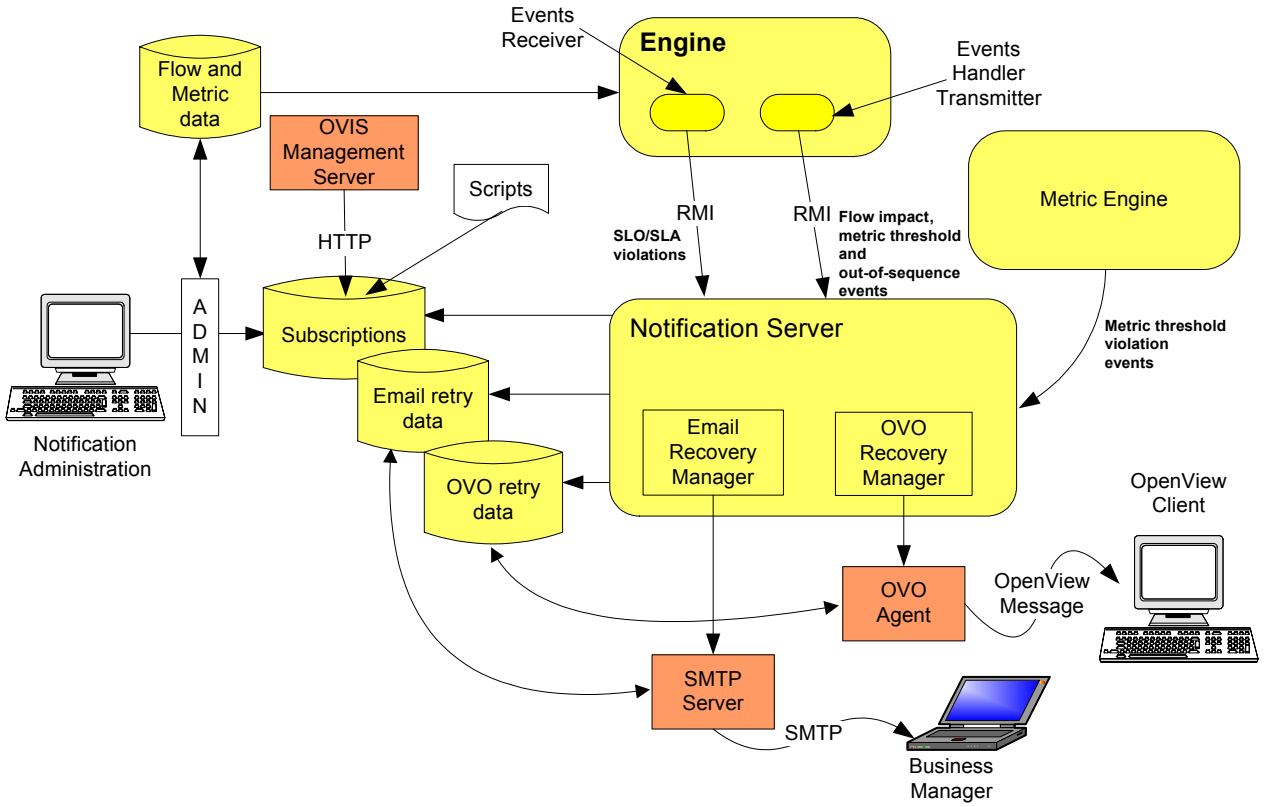

## Notification Server

This component of the OVBPI Server is responsible for notifying you of the flow-impact, out-of-sequence and metric-threshold alerts that you have configured through the Notification Server Administration Console. The Notification Server Administration is described in section Administration Consoles and Interfaces on page 42.

Notifications can be sent either through an SMTP server to an email client, or as an OVO message to an OVO Server. There is a retry mechanism for both types of notifications if either the SMTP server or OpenView Operations is not available for any reason. Notifications are queued and retried after a configurable interval.

Figure 11 shows the architecture of the Notification Server and its relationship to other OVBPI components.

**Figure 11 Notification Server**




Business alerts are created by the Notification Server as a result of receiving:

- flow-impact and out-of-sequence notification events from the business object manager through the Business Event Handler Transmitter.
- SLO and SLA violation notification events directly from the Events Receiver.
- metric threshold violation notification events from the Metric Engine.

The Notification Server sends these events as business alerts to you, based on a set of filters. These are filters that you create and that specify the impact events for which you want notification. You can filter email notifications

based on event name, event type and the name of the business flow. SLAs are filtered according to Customer SLA name, and SLOs are filtered according to Customer, Service Group and SLO name.

In addition to sending alerts when an event is received, the Notification Server can also run any script (`.bat` file) that you have created. Creating scripts is described in the *OVBPI System Administration Guide*.

## Email Notification Messages

The email notifications contain impact information about:

- the business flows that you are monitoring.
- out-of-sequence events.

  Out-of-sequence events are those events that you have chosen to monitor using the `Check sequence` option, which is an option on the OVBPI Modeler.
- SLA/SLO violations.
- metric threshold violations.

There is a default template for these email notifications; however, you can change the layout of the email messages that are sent by the Notification Server. You can also configure the templates for the email notifications and add properties from the business alerts into the messages and configure how many email notifications that you receive for a particular metric threshold violation.

The default templates are provided as part of the OVBPI installation and instructions for creating your own templates and configuring metric thresholds are provided in the *OVBPI System Administration Guide*.

The following are the template types that you can use to change the format of your messages, if required:

- Velocity, which is a Java-based template Engine.
- XSLT, which transforms one XML document into another text document.

It does not matter which tool you use, it is entirely dependent on your own preferences.

### OpenView Operations Messages

OpenView Operations enables you to configure the messages that OpenView components can receive, and also enables you to modify the format of the messages. Refer to the OpenView Operations documentation for details of modifying the format of messages within OpenView and also filtering messages that OpenView components can receive.

By sending OVBPI notification alerts to OpenView Operations, you enable OpenView users to receive these alerts in the clients of their choice. For example, OVBPI alerts can be displayed as Incidents on the Service Desk client GUI. Note that this assumes that Service Desk integration is enabled within OpenView Operations.

Refer to the OpenView Operations documentation to find out how to configure OpenView to send the OpenView messages that are sent from OVBPI to an OpenView management client. For example, the *OpenView Service Desk OpenView Operations Integration Administrator's Guide* provides more information about configuring OpenView Service Desk and OpenView Operations to provide this integration.

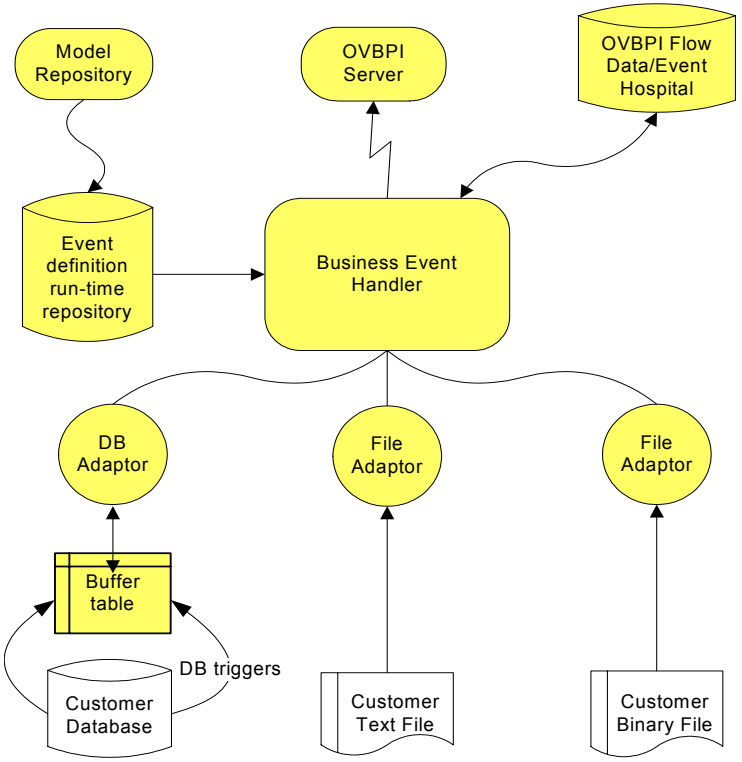
## Business Event Handler

The Business Event Handler provides the Business Impact Engine with normalized business events. Normalized events are events that have a standard (known) set of data attributes that enable it to be recognized by the Business Impact Engine. These events indicate the status changes in the underlying business applications.

Business events provide the data required to progress the flow and also assess the business impact of an operational failure; for example, the value of order or details of a new order that has been created.

The Business Event Handler is built using the openadaptor adaptor framework. openadaptor is an open source adaptor integration toolkit, written in Java, and provides a framework for adaptors. It is a message-based integration toolkit, based on the concept of adaptors providing one-way connections between one or more origins and one or more destinations, including publish/subscribe connections. It can be used to connect systems to systems or systems to middleware.

**Figure 12 Business Event Handler**




In the case of OVBPI, openadaptor is the framework that provides the following OVBPI event adaptors:

- File adaptors

  The flat file adaptor enables you to access information in a binary or ASCII file.

- Database adaptors

  The database adaptor allows you to access information in any JDBC-compliant database. The adaptor uses a polling mechanism to access buffer tables created by database triggers.

- Sockets adaptor

  This is a multi-threaded socket adaptor that by default connects to OVBPI using a specific port.

The Business Event Handler accepts business events received through the adaptors and converts these events into a format that can be accepted by the Business Impact Engine. The Business Event Handler also manages the business events that cannot be immediately delivered to the Business Impact Engine; for example, when the Business Impact Engine has stopped for some reason.

The OVBPI Model Repository holds the design-time Event definitions that you create using the OVBPI Modeler. When these Event definitions are deployed, the Business Events Manager uses the definitions to convert the incoming events into the format required, and to make sure that the correct information is added to the event, so it can be used by the Business Impact Engine.

You can add additional data to events that are received from the individual applications. This allows you to monitor one key application and then assemble the complete set of data required for the event from other applications, rather than monitor all the applications waiting for all the data to become available.

If the Business Impact Engine cannot receive an event from the Business Event Handler for any reason, the Business Event Handler rolls back the event transaction so that it can be retried at a later time. Refer to the *OpenView Business Process Insight Integration Training Guide - Business Events* for more details of event adaptors.

In specific cases, the Business Event Handler places a business event in the Event Hospital. It does this if it receives out-of-sequence events or events that contain errors; an event error might be an event that had missing or corrupt data.

## Event Propagation

This section gives a brief overview of how events that are received through the Business Event Handler are used to progress Business Flows. The *OpenView Business Process Insight Integration Training Guide - Business Events* provides more detail about the Business Event Handler and how to configure it to receive specific business events.

Each time an underlying business system is updated, the Business Event Handler is notified of the change, through a business event, and adds the event details onto the event as defined in the OVBPI Modeler.

The Business Event Handler includes an openadaptor-to-OVBPI adaptor, and all business events for OVBPI are sent to this adaptor. How the adaptor detects the change, depends on how it is configured to communicate with the underlying systems. The adaptor might receive events from a message bus, or it could be notified through a database trigger; see the *OpenView Business Process Insight Integration Training Guide - Business Events* for more information about building and configuring the file and database adaptors.

Where the event mapping is successful, the OVBPI event details are updated and the event is then sent to the Business Impact Engine. If there is no OVBPI event defined that matches the data from the event, the event is moved to the Event Hospital by the adaptor. The business event then must be manually discharged from the Event Hospital so it can be resubmitted to the Business Impact Engine. In the case of an out-of-sequence event, the Business Event Handler places the event in the Event Hospital and marks it for automatic discharge. For other cases where the event cannot be delivered to the Business Impact Engine, the Business Event Handler rolls back the event transaction and returns the event to the source adaptor.

When the Business Impact Engine receives a business event, it determines if the event contains details that apply to any of the data instances that currently exist within the Engine. If there is a Data definition subscribed to the Event, and there are associated data instances, the Business Impact Engine updates the data details using the information from the event and the actions specified through the OVBPI Modeler for the appropriate Event and Data definitions. Any changes to the data cause the start and complete conditions for all relevant flows to be re-evaluated, and where appropriate the flows are progressed.

The Business Impact Engine also determines if there is an impact that needs to be reported for the business flows that use this data, for example, those flows with out-of-sequence nodes.

At the same time, the Business Process Dashboard is polling the OVBPI Flow Data and refreshing its display to show the impacts of the changes.

You need to configure adaptors for each data source (application) from which you want to receive business events. These adaptors provides data for the flow, or flows, that you are monitoring. There are different ways that you can create and configure adaptors, which are described in the *OpenView Business Process Insight Integration Training Guide - Business Events.* You do not need to understand how to create adaptors at this stage, although you do need to understand them when you come to develop your solution.

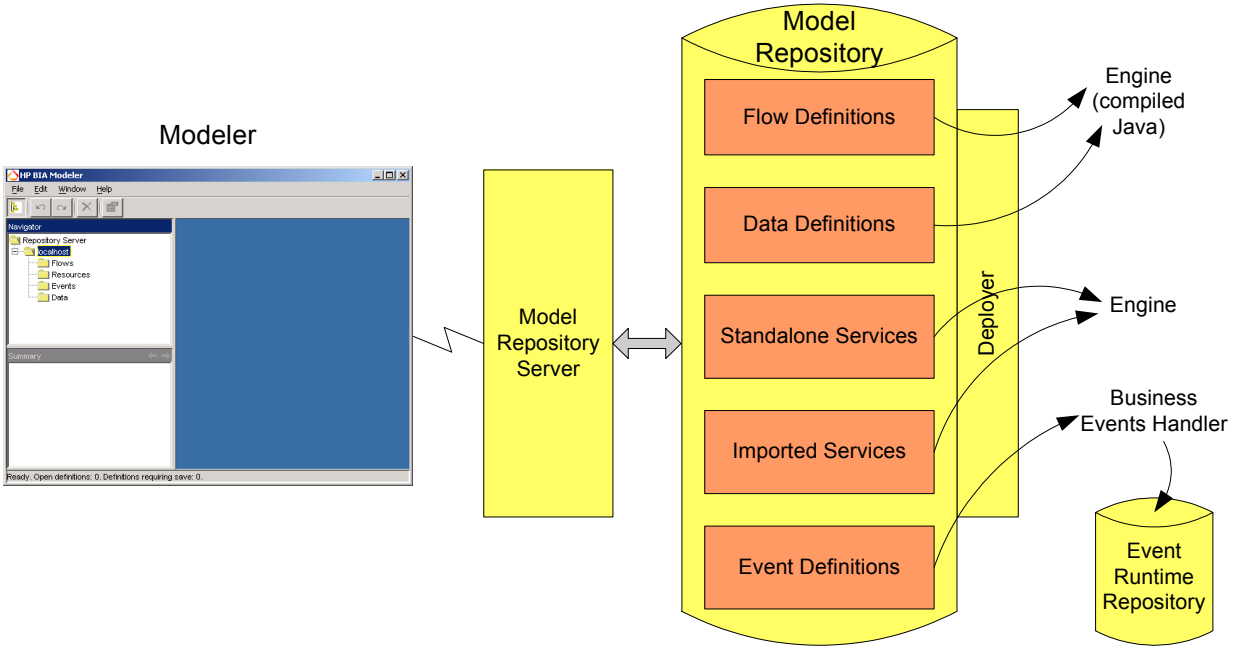
# Modeler

The OVBPI Modeler is a client-side OVBPI component that you use to create your business Flow definitions, Data definitions, Service definitions and Event definitions. It is a graphical tool, with a unified interface for all the definition types that you are creating.

You can create and edit business flows quickly and easily using the OVBPI Modeler, and when you are satisfied that the business definitions are complete, you can deploy them to the Business Impact Engine. You can also import Business Process Execution Language (BPEL) into the Modeler and use the BPEL definition as the start point of your Flow definition.

A local repository cache, in the OVBPI Modeler, holds Flow definitions, including Data, Service and Event definitions until they are saved in the Model Repository, which is a set of database tables. The tables are managed by the Repository Server, which is described in section Repository Server on page 41.

**Figure 13 OVBPI Modeler**



## Repository Server

The Model Repository and the Repository Server are the server-based components that manage the data as it is entered through the Modeler. These components store the model definitions in the database and ensure that the details are consistent and deployable. There is a ToDoList associated with each definition, which reports on the consistency of the definition and keeps track of all the outstanding issues and tasks required to ensure that the definition is consistent and complete. Note that a definition cannot be deployed unless it is consistent.

The deployer component of the Repository Server deploys the different definitions to their destinations as follows:

- Flow definitions, including any imported BPEL definitions, are deployed to the Business Impact Engine as compiled Java code.
- Data definitions are deployed to the Business Impact Engine as compiled Java code.
- Event definitions are deployed to the Business Event Handler to be mapped to events from underlying business applications.
- Service definitions are deployed to the Business Impact Engine as compiled Java code.

The Repository Explorer is a Web-based interface that you can use to view and manage the data stored in the Model Repository.

The OVBPI Modeler and its features are described in more detail in the *OpenView Business Process Insight Integration Training Guide - Modeling Flows*.

# Administration Consoles and Interfaces

The OVBPI components have the following management interfaces that are used for administration and for configuration management:

- OVBPI Administration Console, which is an application that you use to start, stop and configure OVBPI Server Components. Using the OVBPI Administration Console is described in the *OVBPI System Administration Guide*.
- Custom probe dialogs for OVIS. These are Windows interfaces used to configure the OVBPI probes on the OVIS system. Configuring the OVBPI custom probes within OVIS is described in the *OVBPI System Administration Guide*.
- OVBPI Notification Server Administration, which is a Web-based interface (served by Tomcat), to subscribe to the particular events that you want to receive. These events are then sent as email messages, as OpenView Operations messages, or can invoke scripts. The Notification Server Administration Console and how to use it is described in the *OVBPI System Administration Guide*.
- Intervention Client, which is a Web-based interface that enables you to access, or intervene in, business flow instances. Using the Intervention Client is described in the *OVBPI System Administration Guide*.

Later chapters in this document describe these interfaces, and how to use them, in more detail.

# OVBPI Security

You can choose to implement access control for your OVBPI interfaces using HP Select Access or Tomcat (Servlet Engine). When first installed, some of the Web-based interfaces are already configured to use the Tomcat Realm mechanism of authentication; this and how to extend the Tomcat authentication mechanism to all of the OVBPI Web-based interfaces is described in the *OpenView Business Process Insight System Administration Guide*.

HP Select Access is part of the HP Identity Management suite of products and enables you to centralize and automate access control for your OVBPI (and other HP software) components. Select Access is based on a policy-based authentication and authorization mechanism for Web-based applications; for example, for the Repository Browser. There is also customized access control provided for the Modeler using Select Access.

For details of how to configure integration with Select Access, refer to the *HP OpenView Business Process Insight Administration Guide*.

# Where to go Next

You now have an overview of the OVBPI system architecture. The remainder of this guide provides information that you need to configure the components of the OVBPI system.

Read the chapter appropriate to the task that you are trying to complete.

# 3 Integrating with OVBPI

This chapter provides an overview of the possible integration points for your OVBPI system.

OVBPI can integrate with an number of HP OpenView and third-party products. Details of how to configure these integrations is provided in other sections of the OVBPI manuals; this chapter provides an overview of the integration points and not details of how to configure the integration.

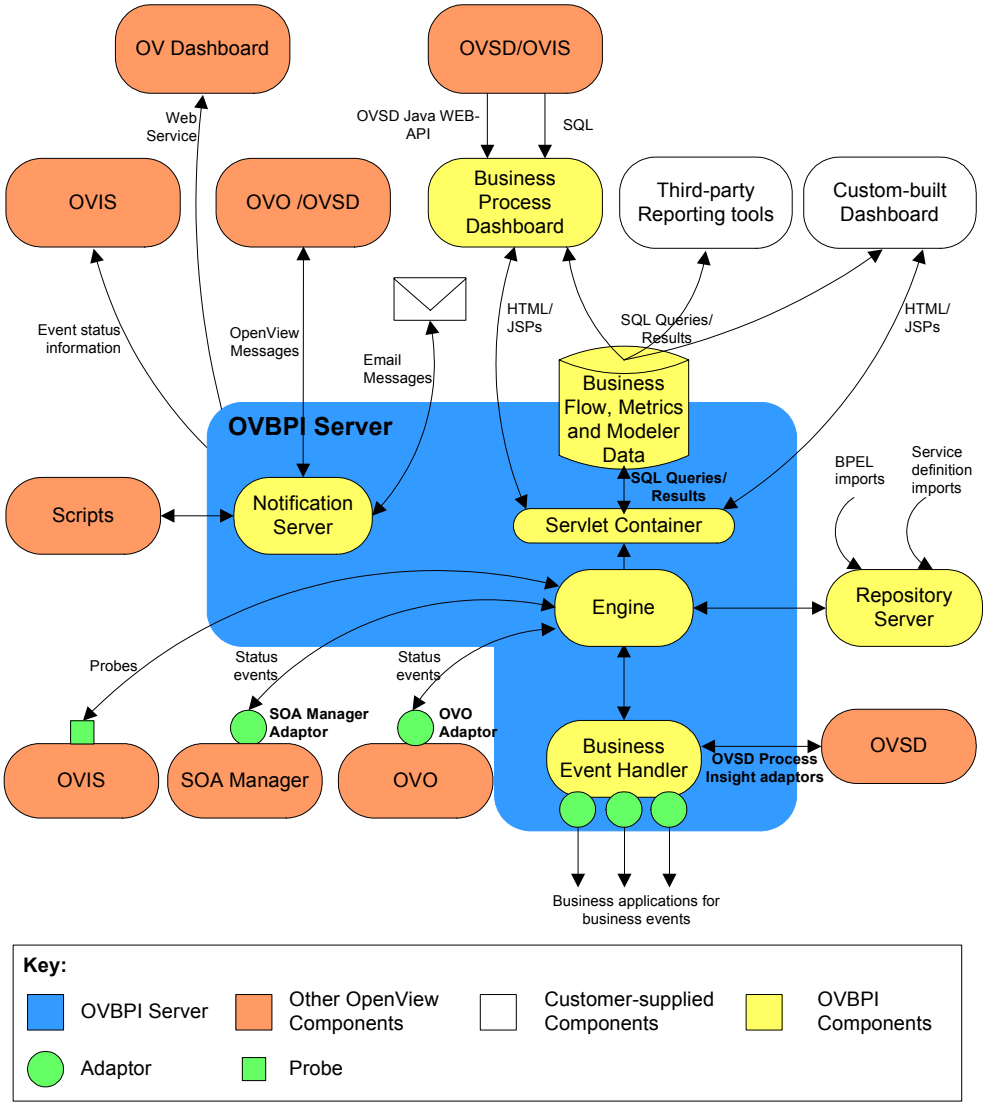
The chapter covers the following topics:


- OVBPI Integration Points on page 46
- OVBPI Adapters on page 50
- iWay Integration on page 56
- OpenView Dashboard on page 57
- Self-Healing Services on page 59

# OVBPI Integration Points

Figure 14 shows a high-level diagram of how components can integrate with OVBPI; these include both design-time and run-time integrations.

**Figure 14 Integrating with OVBPI**



There are a number of different ways that you can integrate with OVBPI:

1. Import service definitions from other OpenView components.

   OVBPI has adapters that you can use to integrate with other OpenView products. You can then link to the operational services reported through these OpenView products from your OVBPI business flows and report on the status of these services and how the services are impacting your business flows. Section OVBPI Adapters on page 50 describes these adapters and the OpenView products that they enable you to integrate with.

2. Configure data sources through the Business Event Handler and import business events from underlying business applications.

   Using the Business Event Handler, and openadaptor technology, you can import business events and use these events to progress your business flows. You can also use these business events to report the business impact where flows are not progressing as they should be. The Business Event Handler is described in Chapter 2, OVBPI Architecture.

3. Configuring notifications through the Notification Server:

   — You can configure email notifications that can be sent when an alert notification event is received by the Notification Server.

   — You can create scripts that are executed when an alert notification event is received by the Notification Server.

   This enables you to integrate the date from alert notifications that are generated by OVBPI into your own spreadsheet applications. Alternatively, you can create a script that generates an SMS notification message.

   — You can configure OVO notifications (OVO messages) that can be sent when an alert notification is received by the Notification Server.

   This enables you to send an OVO message into HP OpenView when a specific event occurs within your business flow. This OVO message can then be tracked or identified by other OVO applications that understand the OVO message format.

   Configuring the Notification Server is described in the *OpenView Business Process Insight Administration Guide*.

4. Using reporting tool, such as Crystal Reports, to report on data in the OVBPI Database.

   You can use the data created by OVBPI in the database to report on your business flows and analyze the behavior of your flows as required. The OVBPI database tables are described in Appendix A, Database Schemas. There is also a contributed document on the distribution media that provides an example of using Crystal Reports to analyze the OVBPI data.

5. Customizing the OVBPI Dashboard and create a new Dashboard for your business flows.

   You can make changes to the OVBPI Dashboard to provide a more closely integrated Dashboard solution for your business flows, you might also want to integrate the data collected by OVBPI with existing customer data from another data source. Refer to the *OpenView Business Process Insight Integration Training Guide - Customizing the Business Process Dashboard* for more details of how you can make changes to your OVBPI Dashboard.

6. Using the OVBPI Dashboard to provide links through to appropriate HP OpenView Service Desk (OVSD) Service Calls and Incident reports for specific flow instances. This is described in detail in Chapter 3, Integrating with OVBPI.

7. Configuring the OVBPI custom probes for OVIS to sample functions of the OVBPI business flows for monitoring purposes.

   OVBPI provides a number of OVIS probes that you can configure to monitor OVBPI business flows, and also report on SLO and SLA violations. This is described in detail in Chapter 4, OVBPI and OVIS.

8. Import BPEL definitions into the OVBPI Model Repository using the OVBPI Modeler.

   When you have imported the BPEL definitions, you can use them as the basis, or starting point for your flow definitions. Refer to the *OpenView Business Process Insight Integration Training Guide - Importing BPEL*, for more information on the options for importing BPEL definitions.

9. Using any of the predefined flows and adapters, which are specifically designed to be used to monitor OVSD processes for the following OVSD modules:

   — OVSD Helpdesk Manager

   — OVSD Change Manager

   The predefined flows, adapters and an example Dashboard are described in the *OpenView Business Process Insight Integration Training Guide - Monitoring Service Desk*.

10. Configure OVBPI to enable the OVBPI Dashboard pages to be navigated by the HP OpenView Dashboard; see section OpenView Dashboard on page 57.

# OVBPI Adapters

OVBPI provides a number of adapters that provide integration points to other HP OpenView products and these adapters are broadly divided into two categories: those that use Web Services as their means of communication and those that do not.

The main function of these adapters is to enable you to use OVBPI to monitor operational events from these OpenView products. Specifically, these are the events that are exposed through the adapter interfaces.

The following are OVBPI adapters that do not use Web Services:

- HP OpenView Internet (OVIS) adapter; see section OVBPI and OVIS Probes and Alarms on page 50.
- HP OpenView Operations (OVO) adapter; see section OVBPI OpenView Operations Adapter on page 52.
- HP OpenView Service Desk adapter; see section HP OpenView Service Desk Adapters on page 53

The following is the first of a series of OVBPI adapters that does use Web Services:

- HP OpenView Service Oriented Architecture Manager (SOA Manager) adapter; see section OVBPI OpenView SOA Manager Adapter on page 54

As the model for creating adapters based on Web Services within OVBPI is extensible, the list of adapters using this model will increase in future versions of the product.

## OVBPI and OVIS Probes and Alarms

OVIS is one of the HP OpenView products that OVBPI can integrate with in order to provide OVBPI users with information on the Internet-related services that OVIS manages. Specifically, the OVIS integration can be used to predict, isolate, diagnose and troubleshoot problems on the services that OVBPI is configured to monitor.

When you have configured a connection from OVBPI to OVIS, the OVBPI Modeler is able to recognize OVIS Services and import these services into the Model Repository. You can then associate these operational services with nodes in your business flows.

OVBPI integrates with OVIS in the following ways:

1. OVBPI polls OVIS services for alarms and violations.

   OVIS measures the availability, response time, setup time, and throughput of specific Internet and network activity. Using the data it receives, OVIS can generate alarms and make them available to OVBPI, and other OpenView products. These alerts and regular information updates keep you informed as to whether or not your Internet and network services are performing efficiently.

   OVIS also generates SLO and SLA violation events, which can be received by OVBPI and sent to the Notification Server, which in turn sends them on as email alerts to anyone who has subscribed to them.

2. OVIS can be configured to probe OVBPI using OVBPI custom probes, which you can install on the OVIS system.

   The Internet Services Manager component of OVIS is responsible for managing data collected by OVIS probes; these probes provide the data that OVIS uses.

   There are three custom probes provided with OVBPI and these probes are used specifically to obtain OVBPI data relating to OVBPI flows and flow instances. The custom probes are installed on the same system as OVIS and are used to calculate:

   — the time taken to progress between two defined nodes in the flow, during the configured time period.

   — the number of flow instances that are currently active at a specified node, the value of the weight parameter for these flow instances and the throughput rate per hour. As an example, the probe might calculate the number of outstanding claims and the value of these outstanding claims at a particular time.

   — the number of flow instances that have been completed throughout the probe period. This provides throughput information for the probe period. The probe also calculates the value of the weight parameter for the flow instances. As an example, the probe might calculate the number of claims that have been completed throughout the probe period and the value of these completed claims. Values returned are presented as an hourly rate.

In addition to the OVBPI custom probes, you can also use OVIS to:

- report on the operational status of your OVBPI system
- define OVBPI-specific SLO and SLAs and report on violations

OVIS probes that report on operational status are created within OVIS as described in the OVIS documentation.

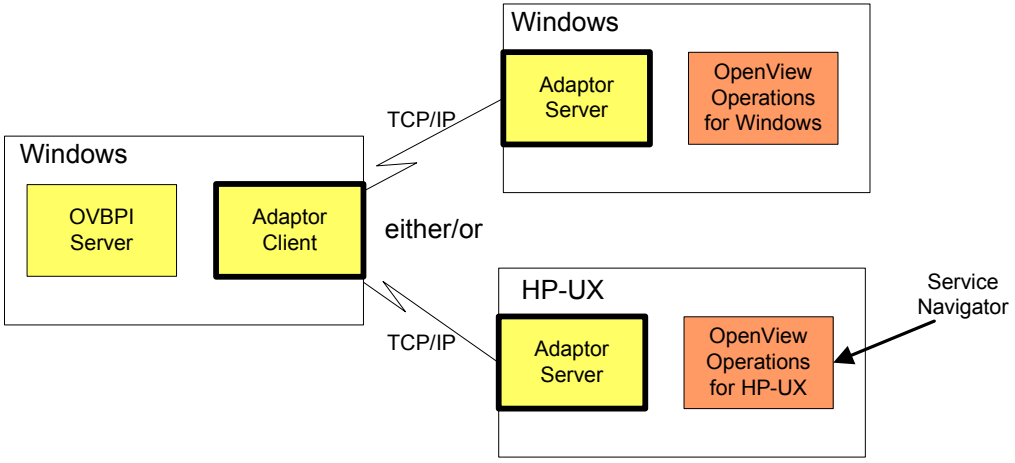
## OVBPI OpenView Operations Adapter

OpenView Operations (OVO) is one of the HP OpenView products that OVBPI can integrate with. OVO can provide status information to OVBPI about the operational services that OVO has been configured to monitor.

The OVBPI OpenView Operations Adapter is responsible for:

- passing OpenView Operations service definitions from either OVSN or OVOW to the OVBPI repository. These services can then be made available through the OVBPI Modeler.
- receiving events from either OVSN or OVOW related to status changes in the services that they are monitoring, and passing them on to the subscribing Flow definitions as OVBPI events.

**Figure 15 OVBPI OpenView Operations Adapter**



An OVBPI Server can be connected to one OVO Server, which can be OVSN or OVOW, it cannot be connected more than one OVO Server. You configure the adapter connection after you have installed OVBPI using the OVBPI Administration Console.

OVBPI service definitions that you enter using the OVBPI Modeler are linked to service definitions made available from OpenView Operations, through the OVBPI OpenView Operations Adapter.

There are two components that make up the adapter:

- The Adapter Client component (referenced as the OpenView Operations Receiver in Figure 15 on page 52), which is installed with the OVBPI Server.
- The Adapter Server component, which is installed as a separate OVBPI component on the system where OpenView Operations is installed. This can be Windows or HP-UX depending on whether you are integrating with OVOW or OVSN.

Details of how to install the OpenView Operations Adapter for OVBPI are provided in the *OpenView Business Process Insight Installation Guide*.

## HP OpenView Service Desk Adapters

HP OpenView Service Desk (OVSD) is one of the HP OpenView products that OVBPI can integrate with. OVBPI provides a number of OVSD adapters that you can use to monitor OVSD ITIL processes.

There is one adapter provided for each of the following ITIL processes:

- Service Calls
- Incidents
- Problems
- Changes
- Work Orders

These adapters are part of OVBPI and can be used with an OVBPI license. Alternatively, if your implementation contains only HP OpenView Service Desk you can use the HP OpenView Service Desk Process Insight license bundle for OVBPI. This license bundle is provided specifically for OVSD integration with OVBPI.

Using these adapters, plus the flow definitions and the customized Dashboard, you can monitor the status of OVSD Change Management and Help Desk activities.

In addition to these adapters, you can configure the OVBPI Dashboard to link through to OVSD Service Calls and Incident reports. These are reports that relate to the impacted instances currently being viewed through the Dashboard.

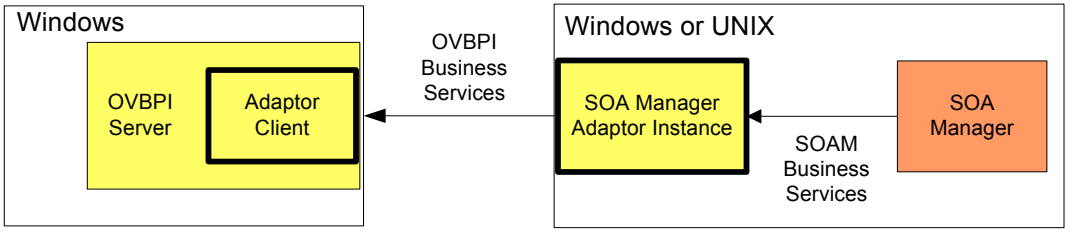
## OVBPI OpenView SOA Manager Adapter

The SOA Manager is one of the HP OpenView Products that you can integrate with. You can configure instances of the OVBPI SOA Manager adapter to integrate with your SOA Manager installations. You can then configure OVBPI to link SOA Manager business events to the business flows that you define.

HP OpenView SOA Manager enables you to manager your service oriented architecture (SOA) resources to ensure their reliability and optimize their performance. The combination of HP OpenView SOA Manager and HP OpenView Business Process Insight provides you with the ability to monitor the health and performance of services running within a Service Oriented Architecture.

In addition, by configuring OVBPI business service metrics and threshold alerts, you can monitor and raise the visibility of architectural and business process performance issues before they have a real impact on the business.

**Figure 16 OVBPI OpenView SOA Manager Adapter**




Details of installing the adapter can be found in the *OpenView Business Process Insight Installation Guide* and more details of how the adapter integrates with OVBPI can be found in Chapter 6, OVBPI and SOA Manager.

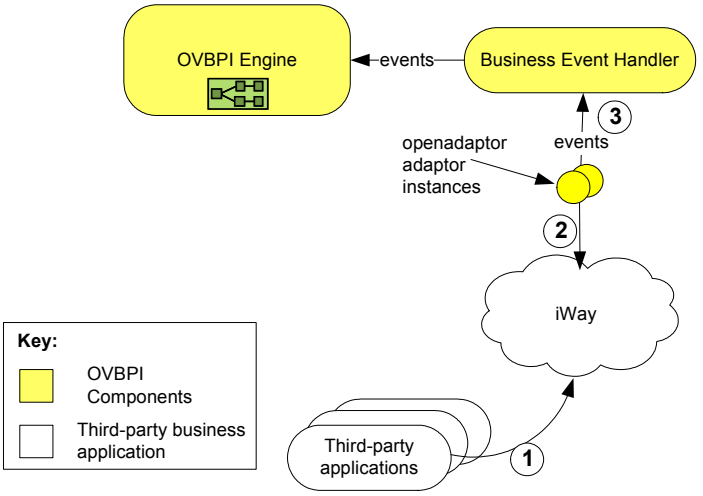
# iWay Integration

The iWay Adaptive Framework (iWay) is similar to openadaptor. Both products address the same requirement, which is to enable developers to quickly and easily integrate components to enable data to flow between heterogeneous data sources.

Both iWay and openadaptor provide the ability to read and write to databases and files; however, iWay provide a larger selection of adapters, particularly for integrating well-known business applications, such as SAP, PeopleSoft and JD Edwards.

The OVBPI integration with iWay is achieved through an openadaptor adapter for iWay, as shown in Figure 17.

**Figure 17 OVBPI/iWay Integration**




From this diagram:

1. Business events are received from third-party applications by iWay.
2. iWay passes the events to openadaptor.
3. openadaptor then passes the events into OVBPI through the Business Event Handler, where they are processed by the Business Impact Engine.

For more information on the iWay integration with OVBPI refer to the *OpenView Business Process Insight Integration Training Guide - Business Events*.

# OpenView Dashboard

The OpenView (OV) Dashboard is an OpenView-wide Dashboard and should not be confused with the OVBPI Business Process Dashboard, which is for use only with OVBPI.

Using OV Dashboard you can quickly and easily create management dashboards that address differing requirements; for example, individual dashboards for service managers, business managers, application managers or operations staff. The content and scope of each Dashboard that you create can be tailored for each user or category of user.

As part of these solutions, you can link the OVBPI service impact information into the OpenView Dashboard; refer to the OpenView Dashboard documentation for more details of the features and functions that it offers and how to add links to the OVBPI Business Process Dashboard.

OVBPI publishes a Web service on its Web Services Provider port. This Web service provides the OV Dashboard with the ability to import service definitions based on deployed OVBPI flows, metrics and thresholds; this information can then be used for building OpenView Dashboard configurations.

The OpenView Dashboard also uses the status information relating to the OVBPI service definitions and metrics, as events are being processed by OVBPI, and presents these results through the OV Dashboard.

When configuring the OV Dashboard access to the OVBPI Web service and to the OVBPI event status information you need to have the following information available:

1. The fully qualified host name of the machine where the OVBPI Server is installed.
2. The port number for the OVBPI Web Services Provider.
3. The port number for the OVBPI Servlet Engine component.
4. The string `OVBPI`, which is required by OpenView Dashboard to identify the connection as an OVBPI connection.

You can find the relevant port numbers using the Administration Console on the Port Numbers page. The OV Dashboard uses this information to build a URL to access the OVBPI Web service and the OVBPI Business Process Dashboard pages.

In addition to this information, you need to:

- Make sure that the OVBPI `Web Services Provider` service is running.

  You start this service from the OVBPI Administration Console. When this service is running, OVBPI exposes its data as a Web Service on the port number used by the Web Services Provider service. On a new installation this is port `44014`. You can check the port number used from the `Port Numbers` page on the Administration Console.

  If the `Web Services Provider` service is not started, the OV Dashboard cannot access any of the OVBPI services, metrics or thresholds.

- Make sure the OV Dashboard is configured to link to OVBPI using the data described earlier in this section. Instructions for configuring OV Dashboard are provided in the OV Dashboard documentation.

When the configuration is complete, OV Dashboard can link directly to the OVBPI business impact data; however, if you have configured the OVBPI Business Process Dashboard to be authenticated by either Select Access or Tomcat (Servlet Engine), this authentication is passed to the OV Dashboard. If you do not want to your OV Dashboard users to have to log on to the OVBPI Business Health pages from within the OV Dashboard, you need to disable the security method that you have configured for the OVBPI Business Process Dashboard; you do this using the OVBPI Administration Console.

In addition, if you have configure OVBPI to integrate with OpenView Service Desk, this information is not available when linking to the OVBPI data from OV Dashboard. However, you can configure OV Dashboard to access this information directly.

# Self-Healing Services

HP Self-Healing Services, as used by OVBPI, automates some of the steps involved in collecting information about a problem and then packaging the information to send it to HP. You can use OVBPI's integration with the Self-Healing Services to gather information about a problem and send this information to HP using a secure Web access mechanism.

Self-Healing Services (SHS) is a free service for HP support customers that automates many of the steps involved in troubleshooting, searching for solutions, and submitting support cases for HP OpenView applications. SHS results in a significant decrease in the amount of time and effort required for you to recognize that a problem occurred, search for documented solutions, submit a support case, collect troubleshooting information, and get a solution.

The type of information that OVBPI collects includes:

- configuration files
- log files
- system environment configuration

The Self-Healing Service is therefore able to reduce the time and effort required to notify HP about a problem with OVBPI.

The *OpenView Business Process Insight Installation Guide* explains how to automatically collect information about a problem using the Self-Healing processes.

The following is an overview of how the Self-Healing Services work:

1. When an error occurs on your system, you can start the Self-Healing Services client and run the OVBPI Data Collector. The Data Collector is specific to OVBPI and provides the Self-Healing Services with the data required to troubleshoot your OVBPI system.
2. The data collected can be securely transmitted to HP support systems using HP's Instant Support Enterprise Edition; this tool is already in use at many customer sites. This enables HP support engineers much faster access to the data they need for solving customer problems. The same data is also available for you to use for troubleshooting if required.
3. The HP support systems analyze your information and publish a system-specific incident report. This report contains information such as:
   - links to potential solution documents
   - patch analysis for OpenView applications
   - configuration analysis.

Details of how to set up your system and download the software that you need to run in order to use the HP OpenView Self-Healing Services client and ISEE client, can be found at the following URL:

**http://support.openview.hp.com/self_healing_downloads.jsp**

There is a `download software` option under the `Self-Healing Services Downloads` heading on this Web page.

# 4 OVBPI and OVIS

This chapter describes how to integrate OVBPI with HP OpenView Internet Services (OVIS). Integration with OVIS includes:

- using OVIS to report on the status of operational services.
- using OVIS to report on SLO and SLA violations.
- configuring the OVBPI custom probes for OVIS. These probes are used to sample functions of the OVBPI business flows for monitoring purposes. You can also set report on SLO and SLA violations resulting from these custom probes.

This chapter describes how to configure the OVBPI custom probes to monitor multiple OVBPI Servers; it does not discuss using OVIS as a source of operational services.

The chapter is structured as follows:

- How OVBPI integrates with OVIS; see section OVBPI and OVIS Integration on page 63.
- Using OVIS to report on the operational status of your OVBPI system; see section Reporting on the Operational Status of Your OVBPI System on page 68.
- Using OVIS to report SLO and SLA violations resulting from changes in the operational status of your OVBPI system; see section Reporting SLO and SLA Violations on page 68. The violations can be reported to the business manager through email using the Notification Server.

  Refer to the *OVBPI System Administration Guide* for details of how to set up the Notification Server to send email notifications concerning these violations.

- Configuring the OVBPI custom probes for OVIS; see section Custom Probes on page 69.
- Configuring the OVBPI custom probes for multiple OVBPI Servers; see section Configuring Probes for Multiple OVBPI Servers on page 87

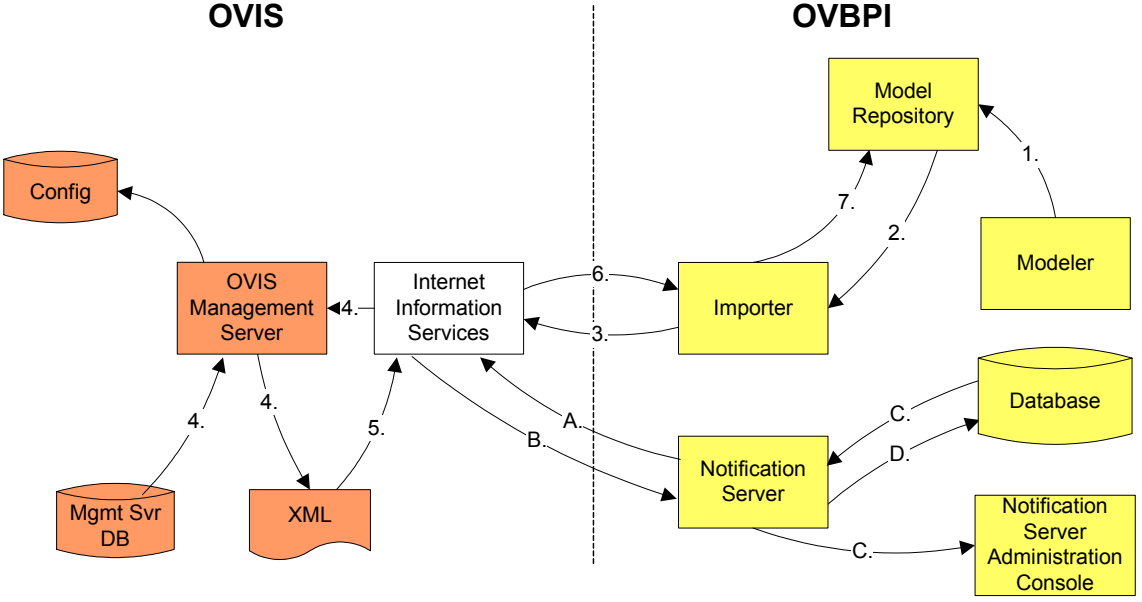
# OVBPI and OVIS Integration

There are two aspects to the OVIS integration with OVBPI: design-time and run-time integration. The integrations are shown in section OVBPI and OVIS Design-Time Integration on page 63 and in section OVBPI and OVIS Run-Time Integration on page 65.

## OVBPI and OVIS Design-Time Integration

Figure 18 on page 63 shows the OVBPI and OVIS design-time configuration.

**Figure 18 OVBPI and OVIS Design-Time Integration**




There are two design-time scenarios shown in Figure 18 on page 63:

- Importing the Service Hierarchy
- Making SLO and SLA details available

## Importing the Service Hierarchy

The first scenario is where the OVIS service hierarchy is imported into the OVBPI Model Repository so the OVIS services can be referenced within business flows:

**Step 1**: User selects the `Link to Services` option within the Modeler to import the OVIS service definitions. The import instruction is sent to the Model Repository. (The Model Repository can also import the OVIS service definitions as a background task, according to its configuration.) Refer to the *OVBPI System Administration Guide* for details of configuring the Model Repository parameters.

**Step 2**: The Model Repository starts the Importer component.

**Step 3**: The Importer component issues an HTTP request to the OVIS Server (through the OVIS Web Server).

**Step 4**: Web Server contacts the OVIS Management Server to obtain the requested service hierarchy from the OVIS database and converts it to an XML document.

**Step 5**: The XML document is sent to the Web Server.

**Step 6**: The XML document is sent by the Web Server using an HTTP response to the Importer component, which parses the XML.

**Step 7**: The parsed XML is stored in the Model Repository and made available to the user through the Modeler interface.

## SLO and SLA Details Made Available

The second scenario is where the SLO/SLA details in the service hierarchy are provided to the Notification Server Administration Console where you can subscribe to those that you want to monitor for potential violations.

**Step A**: Using the Notification Server Administration Console, the user selects the option to refresh the OVIS Service Level configuration information. At this point the Notification Server sends an HTTP request to the OVIS Server (through the OVIS Web Server). The next sequence of steps are identical to those above (steps 4 to 6). The OVIS Service Level configuration information is also refreshed when the Notification Server Administration Console is started.

**Step B**: The XML document containing the service hierarchy is returned to the Notification Server.

**Step C**: The Notification Server parses the service hierarchy (XML document) and displays the available SLOs and SLAs to the user through the Notification Administration Console. Users can then select to subscribe to the SLO and SLA violations.

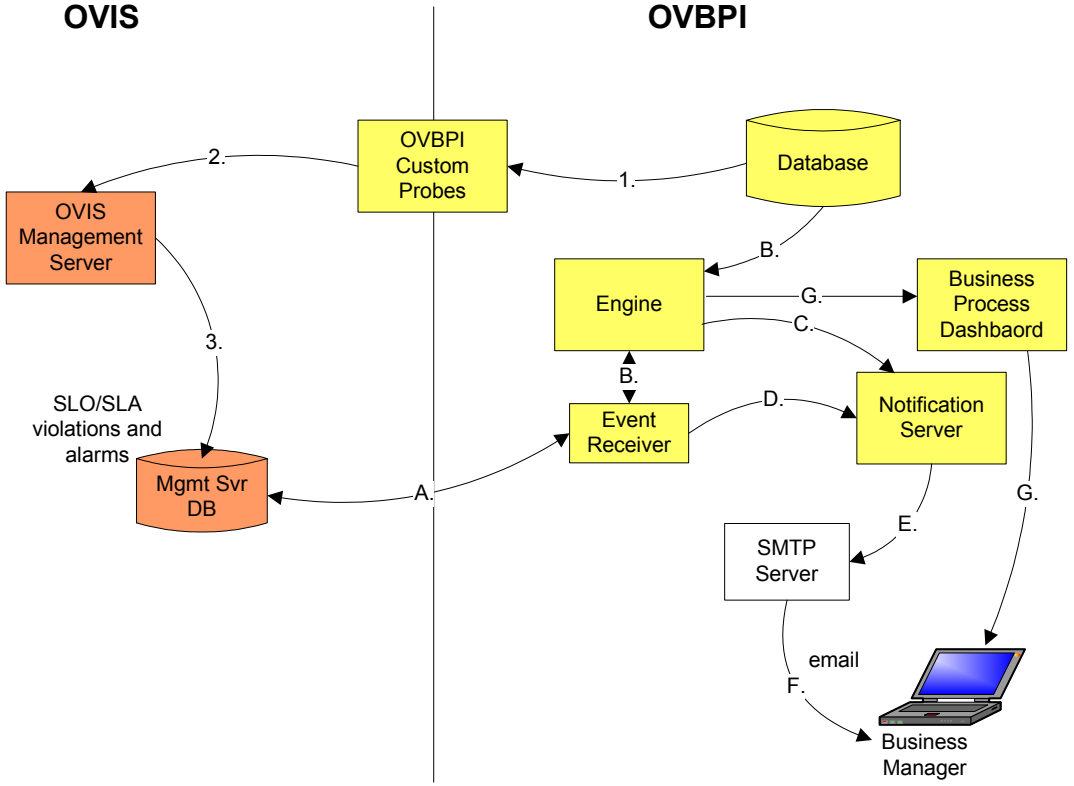
**Step D**: Details of the user subscriptions are stored in the OVBPI database.

Email notifications are delivered as described in the run-time integration; see section OVBPI and OVIS Run-Time Integration on page 65.

## OVBPI and OVIS Run-Time Integration

Figure 19 on page 65 shows the OVBPI and OVIS run-time configuration.

**Figure 19 OVBPI and OVIS Run-Time Integration**



There are three run-time integration scenarios shown in Figure 19.

- Monitoring OVBPI flows using the OVBPI OVIS custom probes
- Receiving flow impact alarms from OVIS
- Sending emails for OVIS Service Level violations

### Monitoring OVBPI Flows Using the OVIS Custom Probes

The following are the steps for monitoring OVBPI, as related to the steps in Figure 19 on page 65:

**Step 1**: The custom probes take their data from the OVBPI database.

**Step 2**: An HTTP message containing the probe data is sent to the OVIS Management Server.

**Step 3**: The probe data is then loaded into the Management Server database in order that it can be measured and used for reporting.

### Receiving Flow Impact Alarms from OVIS

The following are the steps for receiving flow impact alarms, as related to the steps in Figure 19 on page 65:

**Step A:** The OVBPI Event Receiver polls the Management Server database for service impact alarms.

**Step B:** The data on service impact alarms is sent by the Event Receiver to the Business Impact Engine, where the Engine converts the data into internal business events (service impact events). Alarm events are processed by the Business Object Manager component of the Engine and the Engine then updates the status entry in the OVBPI database.

**Step C:** `FLOW_IMPACT` events, resulting from any alarms, are sent to the Notification Server.

**Step E:** The email notifications from the Notification Server are forwarded to your email Server.

**Step F:** The email server delivers the email notifications containing the alerts for service impacts alarms.

**Step G:** The Business Process Dashboard is continually polling the database for status changes and, according to how it is configured, displays the current status of your system.

Note that OVIS writes alarms to its database at specified intervals, and OVBPI is configured to read these alarms at specified intervals. It is therefore possible for the Business Process Dashboard to be reporting different results for an OVIS probe, during this period; for example, the Dashboard might report that the overall OVIS Service status is `Healthy`, but when you link to the OVIS Dashboard from the OVBPI Dashboard, the equivalent OVIS Service might be marked as impacted.

### Sending Emails for OVIS Service Level Violations

The following are the steps for sending email alerts as a result of SLO/SLA violations, as related to the steps in Figure 19 on page 65:

**Step A:** The OVBPI Event Receiver polls the Management Server database for SLO/SLA violations.

**Step D:** The data on any SLO and SLA violations is sent directly to the Notification Server.

**Step E:** The email notifications from the Notification Server are forwarded to your email Server.

**Step F:** The email server delivers the email notifications containing the alerts for SLO and SLA violations.

# Reporting on the Operational Status of Your OVBPI System

You can use OVIS to report on the operational status of your OVBPI system in the same way as you can use OVIS to report on any other component within your business system.

There are no OVBPI-specific configuration considerations, you use the OVIS Configuration Manager to create and configure probes that simulate the use of the services that you want to monitor within OVBPI. These services might simulate the response from the Business Process Dashboard or the database. You use OVIS to configure probes to report the status of your OVBPI system in the same way as you use it to configure probes for other applications.

For information relating to creating OVIS probes to monitor the IT components on which OVBPI depends, refer to the OpenView Internet Service documentation. This chapter does not explain how to set up standard operational alarms within OVIS.

# Reporting SLO and SLA Violations

You can use OVIS to report the OVBPI business flows and their adherence to operational Service Level objectives and conformance to operational Service Level Agreements. Service Level Agreements are created using the OVIS Configuration Manager and can be reported through the OVIS Dashboard.

For information relating to creating SLO and SLA violation alarms using OVIS, refer to the OpenView Internet Service documentation. This chapter does not explain how to set up standard operational alarms within OVIS.

# Custom Probes

In addition to creating operational probes and reporting on SLO and SLA violations, you can use OVIS to report on specific details of your OVBPI flows. OVBPI provides three custom probes for this purpose.

You configure OVIS to monitor OVBPI using the OVIS Configuration Manager. Full details of using the OVIS Configuration Manager are provided in the OVIS documentation. This section describes the OVBPI-specific probe configuration that you need to complete within the OVIS Configuration Manger, following the OVBPI Probes installation.

Before starting the configuration, make sure that you have installed the OVBPI custom probes on the system where the OVIS Management Server is running. Instructions for the installation are provided in the *OpenView Business Process Insight Installation Guide*.

To configure the OVBPI custom probes within OVIS, you first need to create one or more Service Groups within an existing, or newly defined, customer group (`Customers`). A Service Group is where you define a particular service that you want to monitor within OVIS.

OVBPI provides three probes, which enable you to configure specific Monitored Service types within an OVIS Service Group. Within OVIS these Monitored Service types are listed as:

- `C_OVBPI_FLOW - OVBPI_FLOW Probe`

  This probe monitors the following characteristics of flows:

  — The number of active flow instances at the time when the probe is executed.

  — The sum of the values of the weight property for the active flow instances at the time when the probe is executed.

— Throughput values for completed flow instances. The throughput is calculated based on the number of completed flow instances within the configured probe period. The throughput rate is normalized into an hourly rate.

— Throughput values for the weight values for the completed flow instances. The throughput is calculated based on the weight values for the completed flow instances within the configured probe period. The throughput rate is normalized into an hourly rate.

As an example, the probe might calculate the number of claims that were active when the probe was executed, the value of these active claims and the throughput rate of the claims.

- `C_OVBPI_METRIC - OVBPI_METRIC Probe`

  This probe monitors OVBPI business process metric types that you have defined. The probe returns:

  — The time (in seconds) that the most recent business metric took to complete. This is the most recent business metric completed within the configured probe period.

  — The average time taken for all the business metrics to complete within the probe period.

The result of this OVIS metric is reported as `unavailable` if there is no data available. This is because no flow instances have fulfilled the probe requirement in the time interval configured for the probe.

The result of this scenario is that probe data can be intermittently unavailable.

- `C_OVBPI_NODE - OVBPI_NODE Probe`

  This probe monitors the following characteristics of a node:

  — The number of active flow instances at the node at the time when the probe is executed.

  — The sum of the values of the Weight properties for the active flow instances at the node at the time when the probe is executed.

— Throughput values for completed flow instances at the node. The throughout is calculated based in the number of completed flow instances for the node, within the configured probe period. The throughput rate is normalized to an hourly rate.

— Throughout values for the weight values for the completed flow instances at the node. The throughput is calculated based on the weight values for the completed flow instances for the node, within the configured probe period.

As an example, the probe might calculate the number of insurance claims that have been active at a specified node when the probe is executed, the total value of these active claims and the throughput rate of active claims.

The list of Monitored Services includes many services that you can monitor and configure within OVIS. This section covers only the OVBPI-specific services. Refer to the OVIS documentation and OVIS online-help for details of the Monitored Services that are not specific to OVBPI.

Section Creating a New Service Group for an OVBPI Service on page 71 describes the tasks that you need to complete to create a Service Group to define OVBPI-specific Monitored Services. It also describes the information required to create a Service Target for the Service Group, or Service Groups, that you create.

## Creating a New Service Group for an OVBPI Service

To define OVBPI-specific services within OVIS, complete the following steps:

1. Start the OVIS Configuration Manager as follows:

   `Start|Programs|HP OpenView|Internet Services|Configuration Manager`

2. Navigate to the `Customer Name` where you want to add your new service, or create a new `Customer`.

3. Create a new Service Group and select the OVBPI-specific Monitored Service that you want to create. The Monitored Service can be one of:
   - `C_OVBPI_FLOW - OVBPI_FLOW Probe`
   - `C_OVBPI_METRIC - OVBPI_METRIC Probe`
   - `C_OVBPI_NODE - OVBPI_NODE Probe`
4. Provide the information required for the Monitored Service according to the service type that you are configuring as follows:
   - to monitor flow instances across the whole flow, refer to section Flow Instances (C_OVBPI_FLOW) on page 72.
   - to monitor business process metric durations, refer to section Flow Metrics (C_OVBPI_METRIC) on page 74.
   - to monitor flow instances for a specific node, refer to section Node Instances (C_OVBPI_NODE) on page 76.

## Flow Instances (C_OVBPI_FLOW)

This probe obtains flow information from the Business Impact Engine database. It uses this information to calculate:

- The number of active flow instances at the time when the probe is executed.
- The sum of the values of the weight property for the active flow instances at the time when the probe is executed.
- Throughput values for completed flow instances.
- Throughput values for the weight values for the completed flow instances.

The values (or metrics) returned by OVIS are compared with the objectives that you set in the `Objective Information` dialog within OVIS. Each Objective setting defines the expected limits for a specific OVIS metric. These limits are applied to every Service Target in the Service Group. The limits are used to determine when a Service Level Objective is violated and when an alarm event should be generated.

This probe type requires the name of the flow for which instances need to be counted.

When creating a Service Target for this OVBPI Flow Metric, you need to enter the information listed in Table 1 for the Service Target named C_OVBPI_FLOW.

**Table 1 Flow Instances Probe**

| Parameter Name | Description |
|---|---|
| Target Host | The fully qualified host name of the system where the OVBPI database is located. This is the same as the hostname for the system where the OVBPI Server is running. This value must match the hostname in the probes configuration file described in section Configuring Probes for Multiple OVBPI Servers on page 87. |
| Port | This can be left as the default value presented, as it is not used by OVBPI. |
| Username | OVBPI does not use the information entered in this field; however, OVIS does require the value to be null, or a value that is valid for the system where you are using the custom probes. |
| Password | OVBPI does not use the information entered in this field; however, OVIS does require the value to be null, or a value that is valid for the system where you are using the custom probes. |
| OVBPI_Identifier | A unique identifier that you choose, and that must match the unique identifier in the probes configuration file; see section Configuring Probes for Multiple OVBPI Servers on page 87.<br>The default value for the OVBPI_Identifier is OVBPI. |
| OVBPI_DB_User_Name[a] | The OVBPI database username that you provided for OVBPI during the OVBPI installation process. |

**Table 1 Flow Instances Probe**

| Parameter Name | Description |
|---|---|
| OVBPI_DB_Password | The password for the OVBPI database user that you provided for OVBPI during the OVBPI installation process. |
| OVBPI_Flow_Name | The name of the OVBPI flow for which you want to monitor flow instances. |

a. This information can be found through the OVBPI Administration Console under the Oracle Server Access or MS SQL Server Access option, according to your configuration.

## Flow Metrics (C_OVBPI_METRIC)

This probe obtains OVBPI business process metric information from the Metric Views tables in the database. It uses this information to calculate:

- The time (in seconds) that the most recent business metric took to complete.
- The average time taken for all the business metrics to complete within the probe period.

The values (or metrics) returned by OVIS are compared with the objectives that you set in the Objective Information dialog within OVIS. Each Objective setting defines the expected limits for a specific OVIS metric. These limits are applied to every Service Target in the Service Group. The limits are used to determine when a Service Level Objective is violated and when an alarm event should be generated.

The probe requires details of the name of the flow and the name of the associated business process metric defined in the Metric definer in order to monitor this service.

When creating a Service Target for this OVBPI Flow Metric, you need to enter the information listed in Table 2 for the Service Target named C_OVBPI_METRIC.

**Table 2 Flow Metric Probe**

| Parameter Name | Description |
|---|---|
| Target Host | The fully qualified host name of the system where the OVBPI database is located. This is the same as the hostname for the system where the OVBPI Server is running. This value must match the hostname in the probes configuration file described in section Configuring Probes for Multiple OVBPI Servers on page 87. |
| Port | This can be left as the default value presented, as it is not used by OVBPI. |
| Username | OVBPI does not use the information entered in this field; however, OVIS does require the value to be null, or a value that is valid for the system where you are using the custom probes. |
| Password | OVBPI does not use the information entered in this field; however, OVIS does require the value to be null, or a value that is valid for the system where you are using the custom probes. |
| OVBPI_Identifier | A unique identifier that you choose, and that must match the unique identifier in the probes configuration file; see section Configuring Probes for Multiple OVBPI Servers on page 87.<br>The default value for the OVBPI_Identifier is OVBPI. |
| OVBPI_DB_User_Name[a] | The OVBPI database user name that you specified for OVBPI during the OVBPI installation process. |

**Table 2 Flow Metric Probe**

| Parameter Name | Description |
|---|---|
| OVBPI_DB_Password | The password for the OVBPI database user that you provided for OVBPI during the installation process. |
| OVBPI_Flow_Name | The name of the flow that the metric that you want the probe to monitor is associated with. |
| OVBPI_Metric_Name | The name of the business process metric defined for the flow specified in OVBPI_Flow_Name. |

a. This information can be found through the OVBPI Administration Console under the Oracle Server Access or MS SQL Server Access option, according to your configuration.

## Node Instances (C_OVBPI_NODE)

This probe obtains information from the Business Impact Engine database. It uses this information to calculate:

- The number of active flow instances at the node at the time when the probe is executed.
- The sum of the values of the Weight properties for the active flow instances at the node at the time when the probe is executed.
- Throughput values for flow instances that meet the Complete Conditions for the node.
- Throughout values for the weight values for flow instances that meet the Complete Conditions for the node.

The values (or metrics) returned by OVIS are compared with the objectives that you set in the Objective Information dialog within OVIS. Each Objective setting defines the expected limits for a specific OVIS metric. These limits are applied to every Service Target in the Service Group. The limits are used to determine when a Service Level Objective is violated and when an alarm event should be generated.

This probe type requires the name of the flow and the name of the node in the flow.

When creating a Service Target for this OVBPI Flow Metric, you need to enter the information listed in Table 3 for the Service Target named C_OVBPI_NODE_PROBE.

**Table 3 Node Instance Probe**

| Parameter Name | Description |
|---|---|
| Target Host | The fully qualified host name of the system where the OVBPI database is located. This is the same as the hostname for the system where the OVBPI Server is running. This value must match the hostname in the probes configuration file described in section Configuring Probes for Multiple OVBPI Servers on page 87. |
| Port | This can be left as the default value presented, as it is not used by OVBPI. |
| Username | OVBPI does not use the information entered in this field; however, OVIS does require the value to be null, or a value that is valid for the system where you are using the custom probes. |
| Password | OVBPI does not use the information entered in this field; however, OVIS does require the value to be null, or a value that is valid for the system where you are using the custom probes. |
| OVBPI_Identifier | A unique identifier that you choose, and that must match the unique identifier in the probes configuration file; see section Configuring Probes for Multiple OVBPI Servers on page 87.<br>The default value for the OVBPI_Identifier is OVBPI. |
| OVBPI_DB_User_Name[a] | The OVBPI database username that you provided for OVBPI during the OVBPI installation process. |
| OVBPI_DB_Password | The password for the OVBPI database user that you provided for OVBPI during the OVBPI installation process. |

**Table 3 Node Instance Probe**

| Parameter Name | Description |
|---|---|
| OVBPI_Flow_Name | The name of the OVBPI flow for which you want to monitor flow instances. |
| OVBPI_Node_Name | The name of the node within the OVBPI business flow that you are sampling. |

a. This information can be found through the OVBPI Administration Console under the Oracle Server Access or MS SQL Server Access option, according to your configuration.

When you have created a Service Group and created and configured a Service Target for one of these Service Groups, you can continue and define alarms, SLO and SLAs for the probes, according to your requirements.

You can configure multiple Service Targets for a particular Service Group; these Service Targets all have the same Monitored Service. You might want to create multiple Service Targets where a Node (or Nodes) in an OVBPI Flow is dependent on the status of a combination of several OVBPI Flows, Nodes or Metrics (according to probe type). In this case, you can create a Service Group that had several Service Targets, each with different probe properties defined.

You also need to create a new probe location for the Service Targets. In the case of the OVBPI probes, the probe location must be Local System.

# Configuring Alarms and SLOs

This section describes how to create alarms and SLOs for OVIS Monitored Services. These might be OVBPI-specific Monitored Services or they might be other Monitored Services that you have created in order to monitor the operational services on which an OVBPI Flow relies. This section provides an overview of the tasks that you need to complete where they are specific to OVBPI. The OVIS documentation and OVIS online-help provides detailed information about configuring alarms and SLOs, which you should read in addition to the information supplied in this chapter.

## Defining Objective Information for OVBPI Monitored Services

You create Alarms and SLOs for a particular Service Group through the Service Objectives dialog within the OVIS Configuration Manager. Each Service Group that you define has a related Service Target (or Service Targets) and Service Objective. You defined the Service Target in section Creating a New Service Group for an OVBPI Service on page 71.

This section describes the OVIS metrics that are available for the OVBPI custom probes when configuring Alarm and SLO details through the Service Objective dialog.

An OVIS metric is one of a number of values returned by an OVIS probe. As an example, the OVBPI_METRIC_PROBE returns the metrics listed in Table 5 on page 82 each time that the probe is executed.

The OVBPI-specific Service Objective details that you need to supply are specific to each probe type. The required information is described in the following sections:

- C_OVBPI_FLOW_PROBE Objective Information on page 80, for the OVBPI `C_OVBPI_FLOW_PROBE` Monitored Service.
- C_OVBPI_METRIC_PROBE Objective Information on page 82, for the OVBPI `C_OVBPI_METRIC_PROBE` Monitored Service.
- C_OVBPI_NODE_PROBE Objective Information on page 83, for the OVBPI `C_OVBPI_NODE_PROBE` Monitored Service.

## C_OVBPI_FLOW_PROBE Objective Information

When configuring one Service Level Objective for this probe, you can use one of the OVIS metrics listed in Table 4 on page 80. These OVIS metrics relate to active flow instances for the flow.

Note that you can define more than one Alarm, Service Objective and associated OVIS metric.

**Table 4 Service Objective Metrics for C_OVBPI_FLOW_PROBE**

| OVIS Metric | Description |
|---|---|
| AVAILABILTY | Standard OVIS Metric: Set to zero (0) to indicate when no measurement can be retrieved. Set to one (1) to indicate when the service is available. |
| SETUP_TIME | Standard OVIS Metric: Time taken to establish the connection to the OVBPI database. |
| RESPONSE_TIME | Standard OVIS Metric: Time taken for OVBPI Monitored Service to respond. |
| TRANSFER_TPUT | Not used by OVBPI. |
| ACTIVE_INSTANCES_LOW | OVBPI-specific Metric: The lowest number of active flow instances that are acceptable for the defined flow, before the level is considered to be too low. |
| ACTIVE_INSTANCES_HIGH | OVBPI-specific Metric: The highest number of active flow instances that are acceptable for the defined flow, before the level is considered to be too high. |
| ACTIVE_VALUES_LOW | OVBPI-specific Metric: The lowest Weight value for active flow instances that is acceptable for the defined flow, before the level is considered to be too low. |

**Table 4 Service Objective Metrics for C_OVBPI_FLOW_PROBE**

| OVIS Metric | Description |
|---|---|
| ACTIVE_VALUES_HIGH | OVBPI-specific Metric: The highest Weight value for active flow instances that is acceptable for the defined flow, before the level is considered to be too high. |
| INSTANCE_TPUT_LOW | OVBPI-specific Metric: The lowest number of completed flow instances for the configured probe period (throughput) for the defined flow, before the number is considered to be too low. This value is reported as an hourly rate. |
| INSTANCE_TPUT_HIGH | OVBPI-specific Metric: The highest number of completed flow instances over the configured probe period (throughput) for the defined flow, before the number is considered to be too high. This value is reported as an hourly rate. |
| VALUE_TPUT_LOW | OVBPI-specific Metric: The lowest value for the Weight parameter for the completed flow instances over the configured probe period (throughput), before the level is considered to be too low. Shown as an hourly rate. |
| VALUE_TPUT_HIGH | OVBPI-specific Metric: The highest for the Weight parameter for the completed flow instances over the configured probe period (throughput), before the level is considered to be too high. Shown as an hourly rate. |

The remainder of the service level, alarm and objective fields (for example, `Duration`) can be set as they are for all other OVIS Monitored Services according to your business requirements.

## C_OVBPI_METRIC_PROBE Objective Information

When configuring an Alarm or Service Level Objective for the probe, you can use one of the OVIS metrics listed in Table 5 on page 82. You can define more than one Alarm, Service Level Objective and associated OVIS metric.

**Table 5 Service Objective Metrics for C_OVBPI_METRIC_PROBE**

| OVIS Metric | Description |
|---|---|
| AVAILABILTY | Standard OVIS Metric: Zero (0) indicates no measurement can be retrieved, one (1) indicates service is available. |
| SETUP_TIME | Standard OVIS Metric: Time taken to establish the connection to the OVBPI database. |
| RESPONSE_TIME | Standard OVIS Metric: Time taken for OVBPI Monitored Service to respond. |
| TRANSFER_TPUT | Not used by OVBPI. |
| TBN_DURATION_LOW | OVBPI-specific Metric: The lowest duration (in seconds) that is acceptable for the OVBPI business process metric before it is considered to be too low. |
| TBN_DURATION_HIGH | OVBPI-specific Metric: The highest duration (in seconds) that is acceptable for the OVBPI business process metric before it is considered to be too high. |

The remainder of the service level, alarm and objective fields (for example, `Duration`) can be set as they are for all other OVIS Monitored Services according to your business requirements.

## C_OVBPI_NODE_PROBE Objective Information

When configuring one Alarm, or Service Level Objective for an OVBPI business process metric, you can use one of the OVIS business metrics listed in Table 6 on page 83. These OVIS metrics relate to the number of active flow instances at a particular node.

Note that you can define more than one Alarm, Service Objective and associated OVIS metric

**Table 6 Service Objective Metrics for C_OVBPI_NODE_PROBE**

| OVIS Metric | Description |
|---|---|
| AVAILABILTY | Standard OVIS Metric: Zero (0) indicates not measurement can be retrieved, one (1) indicates service is available. |
| SETUP_TIME | Standard OVIS Metric: Time taken to establish the connection to the OVBPI database. |
| RESPONSE_TIME | Standard OVIS Metric: Time taken for OVBPI Monitored Service to respond. |
| TRANSFER_TPUT | Not used by OVBPI. |
| ACTIVE_INSTANCES_LOW | OVBPI-specific Metric: The lowest number of active flow instances that are acceptable at the node specified, before the level is considered to be too low. |
| ACTIVE_INSTANCES_HIGH | OVBPI-specific Metric: The highest number of active flow instances that are acceptable at the node specified, before the level is considered to be too high. |
| ACTIVE_VALUES_LOW | OVBPI-specific Metric: The lowest Weight value for active flow instances that are acceptable at the node specified, before the level is considered to be too low. |

**Table 6 Service Objective Metrics for C_OVBPI_NODE_PROBE**

| OVIS Metric | Description |
|---|---|
| ACTIVE_VALUES_HIGH | OVBPI-specific Metric: The highest Weight value for active flow instances that are acceptable at the defined node, before the level is considered to be too high. |
| INSTANCE_TPUT_LOW | OVBPI-specific Metric: The lowest number of completed flow instances at the node specified (over the configured probe period), before the number is considered to be too low. This is a throughput and the parameter is given as an hourly rate. |
| INSTANCE_TPUT_HIGH | OVBPI-specific Metric: The highest number of completed flow instances at the node specified (over the configured probe period), before the number is considered to be too high. This is a throughput and the parameter is given as an hourly rate. |
| VALUE_TPUT_LOW | OVBPI-specific Metric: The lowest value for the Weight parameter for completed flow instances at the node specified (over the configured probe period), before the level is considered to be too low. Shown as an hourly rate. |
| VALUE_TPUT_HIGH | OVBPI-specific Metric: The highest value for the Weight parameter for completed flow instances at the node specified (over the configured probe period), before the level is considered to be too high. Shown as an hourly rate. |

The remainder of the service level, alarm and objective fields (for example, `Duration`) can be set as they are for all other OVIS Monitored Services according to your business requirements.

## Defining Service Level Agreements for OVBPI Monitored Services

You create an SLA for a particular Service Group through the Service Agreements option for a particular `Customer` within the OVIS Configuration Manager.

A Service Level Agreement can be created based on the results of one or more Service Objectives that are defined for the same Customer.

There are no OVBPI-specific SLA options. You can create SLAs for your OVBPI system using the OVIS options offered through the Service Level Agreements configuration. To access this configuration option, you need to create a new Service Agreement from the OVIS Configuration Manager.

You can also configure users to be alerted to SLA violations using the Notification Server.

# Making Sure OVIS Adds Alarm Data to Its Database Tables

Within OVIS there is an option to configure whether or not OVIS enters alarm data to its database, specifically to the database table: `IOPS_ALARM_DATA2`. This option is the `Event DB` or `Database` option on the `Configure Alarm Destinations` dialog. It is the same option as is required to enable NNM integration.

If this option is disabled, no alarm data is written to the table and because OVBPI polls the OVIS database, it cannot therefore report on the OVIS alarms.

You need to make sure that this option is set within the OVIS configuration in order for the integration between OVBPI and OVIS to be successful. Use the OVIS Configuration Manager to access the `Alarm Destinations` option. The option is accessed from the `File|Configure|Alarm Destinations` menu.

# Defining Probe Locations

The `Probe Location Info` dialog is the same for all the OVIS probes. You can therefore use the Probe Location Info dialog to configure details of the location of the probes that have been installed for OVBPI.

OVBPI supports local probes only. Therefore, when creating a probe location for an OVBPI probe, you must select `Local System` as the value for the `Probe Location` parameter. The probes are installed on the same system as the OVIS Configuration Manager.

# Configuring Probes for Multiple OVBPI Servers

If you have installed more than one OVBPI Server, each using different databases, and you want each OVBPI Server monitored by the same OVIS server, you need to modify the configuration file for the OVBPI custom probes. This file is located at:

*OVIS-install-dir*\probes\OvbpiProbe.cfg

You can install multiple OVBPI systems (or Servers) within your organization; however, the Servers cannot share business flows, or business flow data in the OVBPI database, they are completely independent implementations.

There is an example format of a section that is used to configure a Microsoft SQL Server database and there is an example format for an Oracle database. You can have as many sections of either database file section type defined in OvbpiProbe.cfg.

The configuration file for the custom probes contains one, or more, of the following sections, where each section starts with a unique identifier:

```
[OVBPI-identifier]
RDBMS_TYPE=SQL Server
ODBC_DRIVER_NAME=SQL Server
OVBPI_DB_SCHEMA_NAME=schema-name
OVBPI_DB_SERVER_HOSTNAME=db-instance-name
```

where:

- *OVBPI-identifier* is a unique identifier that you choose to distinguish this section from other sections in the configuration file.
- SQL Server is the string that you include when you are configuring the custom probes for an OVBPI Server using an RDBMS database type of Microsoft SQL Server database.
- *SQL Server* is the SQL Server ODBC driver name that has been configured for your system. You can find out what it is from:

  Start|Programs|Control Panel|Administrative Tools|Data Sources (ODBC)

- *schema-name* is the name of the database schema that you configured for the OVBPI data in the database specified by *db-instance-name*.
- *db-instance-name* is the name of the database where the OVIS data for OVBPI is located; for example:

  ```
  hostname
  hostname\qualifier
  ```

  The SQL Server database name can be the same as the hostname (including `localhost`) or it can be a combination of hostname and a string that you specify.

```
[OVBPI-identifier]
RDBMS_TYPE=Oracle
ODBC_DRIVER_NAME=Oracle ODBC Driver
ORACLE_NET_SERVICE_NAME=tns-server
```

where:

- *OVBPI-identifier* is a unique identifier that you choose to distinguish this section from other sections in the configuration file.
- `Oracle` is the string that you include when you are configuring the custom probes for an OVBPI Server using an RDBMS database type of Oracle.
- *Oracle ODBC Driver* is the Oracle ODBC driver name that has been configured for your system. You can find out what it is from:

  ```
  Start|Programs|Control Panel|Administrative Tools|Data
  Sources (ODBC)
  ```

- *tns-server* is the TNS name configured in the Oracle file `TNSNAMES.ORA` for the Oracle database that the OVBPI Server is configured for.

The following is an example of the file following an installation of the OVBPI custom probes on the same system as the OVBPI Server, where the OVBPI is configured to use a Microsoft SQL Server database:

```
[OVBPI]
RDBMS_TYPE=SQL Server
ODBC_DRIVER_NAME=SQL Server
OVBPI_DB_SCHEMA_NAME=OvbpiSchema
OVBPI_DB_SERVER_HOSTNAME=host1\sql-svr-2005
```

# 5 OVBPI and OVSD

OVBPI integrates with OVSD in a number of ways:

- The OVBPI Dashboard can provide links through to appropriate HP OpenView Service Desk (OVSD) Service Calls and Incident reports for impacted instances.
- Through a set of predefined flows, adapters and a customized Dashboard, which can be used to monitor OVSD processes; specifically, processes for the following OVSD modules:
  - OVSD Helpdesk Manager
  - OVSD Change Manager
- The OVBPI Notification Server can send OVO Messages to an OVO system for delivery to OVO components, including OVSD.

This chapter focuses on the OVSD integration through the Dashboard for Incident reports and Service Calls. For information about the predefined flows and adapters available for monitoring OVSD processes, refer to the *OpenView Business Process Insight Integration Training Guide - Monitoring Service Desk*. For information about OVO notification messages, refer to the *OVBPI System Administration Guide* and the *OpenView Business Process Insight Concepts Guide*.

# OVBPI and OVSD Service Call and Incident Information

One of the features of HP OpenView Service Desk (OVSD) is to provide information about Service Calls and Incidents. A Service Call is a record of a request from a user for support for an IT service. An Incident is an operational event that is not part of the standard operation of the IT system. Both service calls and Incidents relate to operational services that are defined within your OpenView IT implementation. These can be OVO and OVIS services.

Within OVSD, service items contain information about services that form a particular IT system; these service items can be associated with HP OpenView Operations services. Service items can also be associated with Service Level Agreements, which in turn can be monitored using OVIS.

OVBPI integrates with both OVO and OVIS (at the service group level) for operational service information and, in the case of OVIS, for OVIS metrics definitions.

The information provided through the OVBPI Dashboard's integration with OVSD provides the OVSD context for the services that are being monitored and tracked through the dashboard. For example, if a service is not available for any reason, the OVSD Service Call and Incident information indicates whether the problem is being addressed, who is responsible for resolving the problem, and how many users are affected.

OVBPI integrates with OVSD through the OVBPI Business Process Dashboard. This integration is achieved through the OVSD WEB-API, which is a Java interface used to access the OVSD data. This chapter describes the two main integration options and how you configure both the OVBPI Server and OVSD to achieve the integration that you require.When integrated, the appropriate Service Call and Incident information for a service is shown through the OVBPI Business Process Dashboard.

The following sections describe how you configure OVBPI to integrate with OVSD. The sections assume that you understand how to use OVSD and its management interface.

## OVSD Web API

The OVBPI Business Process Dashboard obtains information on Service Calls and Incident reports from OVSD using the OVSD Web API, specifically, using the Java web archive, `web-api.jar`. Both OVBPI and OVSD must use the same version of this file and OVSD changes this `.jar` file with every release, including service pack releases. Therefore, if you are integrating with a version of OVSD that is not the version (including patch version) specified in the *OpenView Business Process Insight Installation Guide*, you need to copy the file `web-api.jar` from the OVSD system to the OVBPI system. The file is located in the `api` folder of the appropriate CD-ROM, or disk image, within the Service Desk distribution media.

Before copying this file to the OVBPI system, you need to shut down all the OVBPI Server components, including the Administration Console. This ensures that none of the OVBPI components are using the file before you copy it. When all the components are shut down, locate the `web-api.jar` file on the OVSD system and copy it to the following locations under your OVBPI installation directory:

- *ovbpi-install-dir*\java
- *ovbpi-install-dir*\nonOV\jakarta-tomcat-5.0.19\webapps\ovbpidashboard2-10\WEB-INF\lib

  The Business Process Dashboard needs to have been started following a new installation for this directory to be populated.
- *ovbpi-install-dir*\examples\bia\BusinessProcessDashboard\WEB-INF\lib

## Automatic OVSD Service Mapping

OVBPI can integrate with OVSD by attempting to map the OVO or OVIS services defined within the OVBPI Modeler to the same services defined within OVSD.

For this integration method to be successful:

- OVSD must have imported OVO services from the same OVO server as OVBPI (OVOW or OVSN).
- OVIS service level agreements (SLAs) must have been defined within OVSD and then exported from OVSD into OVIS.
- From the Service Desk Client, use the SLM option to add a new Service and then add the OVO service name to the `SN Name` field.

If the OVO or OVIS service is not created in this way, OVBPI is unable to display the OVSD data through the OVBPI Dashboard. If this is the case, you can define a custom field within OVSD in order to integrate OVBPI and OVSD. You also need to define custom fields if you want to report on Standalone services that have been defined within OVBPI. Section Defining OVSD Custom Fields on page 92 describes how to create a custom field for OVBPI, within OVSD, and the data that you need to enter in the custom field.

For details of how to configure OVBPI to interoperate with OVSD automatically, refer to the *OVBPI System Administration Guide*.

## Defining OVSD Custom Fields

An alternative method of integrating OVBPI services with OVSD services is to define Custom Fields within OVSD. These Custom Fields can then be used to hold the names of OVBPI services. You need to do this if OVO and OVIS services have not been created in OVSD as described in section Automatic OVSD Service Mapping on page 92.

You might also want to create custom fields to enable Service Call and Incident information to be reported on Standalone services that are defined within OVBPI.

OVBPI supports a number of OVSD custom fields that you can use to identify OVBPI services; these are OVSD Service fields. The Custom Fields that are available for you to use within OVSD are `Srv.Text1` through to `Srv.Text5`.

In order to use this method of integration, you first need to set up an appropriate custom field within OVSD. You can then select the custom field from the OVBPI Administration Console to complete the integration.

The following steps outline what you need to do within OVSD to configure a custom field for use with OVBPI:

1. Make sure that you are logged into OVSD from an account that has administrator permissions.
2. Start the Administrator Console and select `System...` from the `Tools` menu within OVSD
3. Navigate to the Custom Fields dialog. For example, select:

   `hp OpenView service desk > Data > Custom Fields`

   This is where you can select, rename and activate custom fields within OVSD
4. Open the `Service` option within the `Custom Fields` dialog.

   In order to integrate with OVBPI services, you need to define a custom field of the type `Text 255` for the Service option within OVSD.
5. From the `Service - Custom Fields` dialog, select one of the supported Fields that is currently not activated. This is one of `Srv.Text1` through to `Srv.Text5`.

   If all the fields are already activated, you need to find out if any are now redundant and can therefore be re-used for OVBPI. If none of the supported fields is available, this method of integration cannot be used.
6. If one of the Custom Fields for the Service is available, rename the field to something that is meaningful for your implementation, for example, OVBPI Service. The name needs to be meaningful to the OVSD user who will configure the Service within OVSD as part of the integration with OVBPI.
7. Check the `Activate` check box to enable to enable the custom field.

You have now completed the steps to create a new custom field within OVSD. The next stage is to add the new custom field to the OVSD Service dialog, so it is available to the user to add configuration data.

To add the new custom field to the Service dialog, complete the following steps:

1. Make sure that the Administration Console is active.
2. Navigate to the Forms Designer tool for services, for example:

   `hp OpenView service desk > Presentation > Forms > Service`

   The list of forms that can be modified is listed in the right-hand pane.
3. Open the Service forms designer window.

   The Form Designer dialog showing the current structure of the OVSD Service dialog is displayed.
4. Select the newly created custom field from the Attributes menu and drag it onto the Forms Designer dialog in the position where you want it to appear.
5. Save your changes using the `File|Save` option.

This method of updating the OVSD Service dialog updates all the Service dialogs within OVSD. You can be more selective when you modify forms; refer to the HP OpenView Service Desk documentation for full details of customizing OVSD forms.

When you have completed this configuration step, any OVSD service can be linked to an OVBPI Service using the custom field.

The following steps describe how you link an OVBPI Service using this newly created field:

1. From within the OVSD desktop, select the option to create a new service, for example:

   `File|New|Service`
2. From the `Choose Template` dialog, select the template that you want to use to create the service.
3. From the `New Service` dialog, complete the details of the new service that you are creating, and in the new custom field that you have created for the form, you need to enter the Service name for the OVBPI Service that you want OVSD to report on.

   The format of the OVBPI Service name entered within the OVSD must be the Service name as it appears within the OVBPI Modeler, excluding the OVBPI-specific prefix.

The following is an example of a Service Name hierarchy that might be defined within the OVBPI Modeler for an OVO service:

```
Model Repository
    OVO Services
      CRM Application
```

In this example, you enter `CRM Application` into the new custom field. You do not include the `OVO Services` prefix.

The following is an example of a Service Name hierarchy that might be defined for OVIS:

```
Model Repository
    OVIS Services
      Insurance
          CRM Application
```

In this example, you enter `Insurance/CRM Application` into the new custom field. You do not include the `OVIS Services` prefix.

The following is an example of a Service Name hierarchy that might be defined for a Standalone Service:

```
Model Repository
    Standalone Services
          My Service
```

In this example, you enter `My Service` into the new custom field. You do not include the `Standalone Service` prefix.

4. Save the new OVSD service.

   This service is now in a form that can be accessed by OVBPI. When an OVSD service definition is created within OVBPI, and subsequently mapped to an OVSD service as defined in OVSD, the OVBPI Dashboard is automatically notified of the Incident report. As a result, you can link from the impacted Service to the Incident report through the Dashboard.

The final step is to configure OVBPI to integrate with OVSD using the newly created custom field.

When you set up the OVSD integration with OVBPI, you need to create an OVSD user account for OVBPI to use. You are strongly advised to create a user account specifically for OVBPI with the following characteristics:

- the user account should allow for concurrent users
- the user account should have the role `Helpdesk`

You provide details of this account when you set up the `OVSD Interoperability` through the OVBPI Administration Console or through the OVBPI installation.

# 6 OVBPI and SOA Manager

This chapter describes how to integrate OVBPI with HP OpenView SOA Manager, specifically to enable OVBPI to report on the status of SOA Manager business services.

OVBPI can also integrate with SOA Manager and use SOA Manager as a source of business events. This is described in the *OpenView Business Process Insight Integration Training Guide - Business Events*.

The chapter is structured as follows:

- How OVBPI integrates with SOA Manager; see section SOA Manager and OVBPI Integration on page 98.
- Obtaining status information from SOA Manager; see section Business Events and SOA Manager on page 101.
- Configuring the OVBPI SOA Manager Adapter; see section Configuring Access to the SOA Manager Adapter on page 102.

# SOA Manager and OVBPI Integration

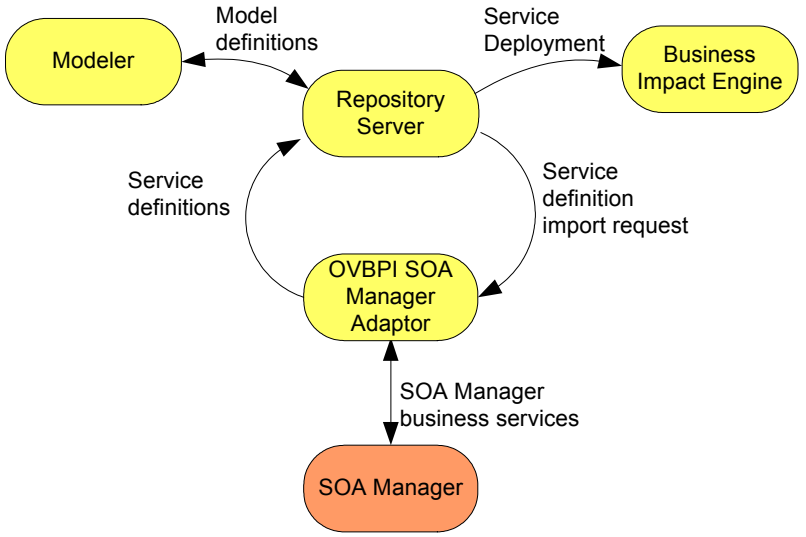
HP OpenView SOA Manager enables you to manage your service oriented architecture (SOA) resources to ensure their reliability and optimize their performance. It is a model-based management solution for managing a SOA in the context of understanding the health of the business services that the SOA architecture supports.

There are two integration points between OVBPI and SOA Manager:

1. The first is the ability to use the SOA Manager service model to provide the status of SOA Manager business services to OVBPI. This integration is through the OVBPI OpenView SOA Manager Adapter as shown in Figure 20 on page 99. Configuring the OpenView SOA Manager Adapter is described in section Configuring Access to the SOA Manager Adapter on page 102.
2. The second is where OVBPI can receive business events from SOA Manager for the OVBPI business flows that it is monitoring. This integration is described in section Business Events and SOA Manager on page 101. The details of how to configure this integration, through the Business Event Handler, is described in the *OpenView Business Process Insight Integration Training Guide - Business Events*.

Figure 20 shows how the SOA Manager Adapter enables you to link to SOA Manager business services and make them available to OVBPI business flows through the Repository Server.

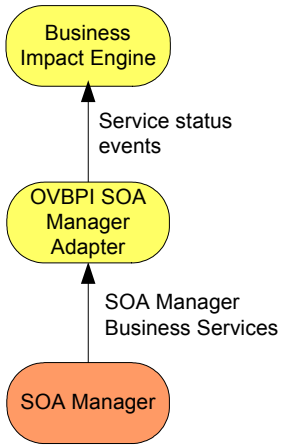
**Figure 20 OVBPI and SOA Manager Business Services**




When you have configured the SOA Manager Adapter, you can import the required SOA Manager business services and link them to your business flows. If you do not configure the integration, you are unable to use the OVBPI Modeler to link to these business services or synchronize with them.

You then deploy the business flow and the services are deployed to the Business Impact Engine to be monitored; see Figure 21.

**Figure 21 OVBPI SOA Manager Adapter - Business Service Deployment**




The framework for the SOA Manager Adapter is installed as part of OVBPI. You create an instance of the SOA Manager Adapter for each SOA Manager system that you are monitoring from the OVBPI Administration Console.

OVBPI service definitions that you enter using the OVBPI Modeler are linked to business service definitions made available from OpenView SOA Manager, through the OVBPI SOA Manager Adapter.

At run time, status changes for the business services that you have subscribed to in your flows are monitored and managed by the Business Impact Engine.

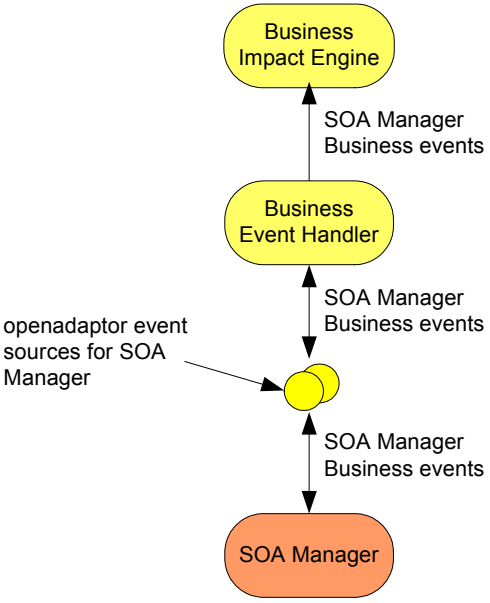
The OVO SOA Manager adapter polls SOA Manager for updates to the business services that it is reporting on. These are the services that you have linked to through the OVBPI Modeler.

# Business Events and SOA Manager

You can also configure your OVBPI system to receive SOA business events by configuring an event source for SOA Manager using openadaptor; see Figure 22. An event source specifically for SOA Manager is installed with your OVBPI Server.

You need to complete the openadaptor configuration and link it to the OVBPI event sink as described in the *HP OpenView Business Process Insight Training Guide - Business Events*.

**Figure 22 OVBPI and SOA Manager Business Events**



# Configuring Access to the SOA Manager Adapter

Make sure that you have installed the adapter and created an instance of the adapter on the target machine; refer to the *OpenView Business Process Insight Installation Guide* for details of installing and starting the adapter.

When you have installed and configured an instance of the adapter, you need to configure the integration (service source) between OVBPI and the SOA Manager adapter using the Administration Console as follows:

1. Select `Operational Service Sources` from the Navigation tree in the Administration Console.

   The right-hand pane shows a list of Operational Service Sources.

2. From the right-hand pane, select the `Add` button to add a new Service Source for the SOA Manager adapter instance that you have created.

   You are presented with the `Service Oriented Architecture Manager Source Properties` dialog.

3. Enter values for the properties of the SOA Manager Service Source. The properties are fully described in the *OpenView Business Process Insight Administration Guide*:

   a. `Service Source Name`

   b. `Description` (Optional)

   c. `Product Name`, which is set as Service Oriented Architecture Manager. The interface allows for additional Service Sources to be added in future versions of OVBPI.

   d. `Hostname`

   e. `Port`

   f. `Status Event Poll Interval`

   In addition, you can configure a Web Proxy for your Web Services connection if you have one.

4. Click the `OK` button when you modifications are complete.

5. Make sure that the `Enabled` check box is selected in the column next to the new service source entry.
6. Click the `Apply` button to apply your changes
7. Move to the `Component Status` screen and stop and restart all the OVBPI components.

The configuration is now complete and you can access SOA Manager business services from OVBPI.

# A Database Schemas

This appendix details the database schemas that are defined to generate reports from the OVBPI impact data and for use by the Business Impact Engine, Metric Engine and Business Events Handler.

The schemas are described so you can use the information generated for your own reporting purposes. Do not modify any of the data in these database tables as OVBPI relies on the data being internally consistent in order to operate. Changing the values in the database tables will impact the behavior and operation of your OVBPI system.

A schema describes the logical structure of the OVBPI database and defines the tables, fields, indexes and views. It also shows the relationship between the fields and the tables.

The Flow schema, Business Event Handler schema and the Business Entity schema are optimized for the application processing. The Metrics schema is optimized for reporting and queries.

You can use the schemas to provide the information that you need to access flow data that you want to incorporate into your reports, or that you want to incorporate into a dashboard that you are developing.

This appendix describes the following schemas:

- Business Metrics schema, which is populated using data generated by the Metric Engine. This is historical data collected as a result of business process metrics that you configure using the Metric definer. Section Business Metrics Schema on page 108 describes this schema.
- Flow schema, which is a static schema created as a result of a flow being defined and subsequently deployed using the OVBPI Modeler; see section Flow Schema on page 139.

- Business Entity Schema, which is directly affected by the users' modeling and reflects the business entities modeled by the user; see section Business Entity Schema on page 157.
- Business Event Handler schema, which describes the Event Hospital and the hospital tables in the database. Out-of-Sequence events and events that contain errors, or that are corrupt, are placed in the Event Hospital. Section Business Event Handler Schemas on page 161 describes this schema.

# Building Applications to Use the OVBPI Metrics Data with Microsoft SQL Server

You can use the data in the OVBPI schemas in your own reporting applications, for example, for a custom-built business dashboard.

If you intend to use Microsoft SQL Server and write SQL select statements that access more than one database table, you are advised to set the priority for database access to be low for applications accessing this data using the following SQL statement:

```
set deadlock_priority low;
```

You also need to design your application to retry if it fails to access the data.

This prevents deadlock situations occurring in cases where an application and the Business Impact Engine are accessing database tables at the same time. In this case, the application might initially fail to access the data; however, it will subsequently retry.

By setting the access priority to low for applications, the Business Impact Engine has priority and its performance is not impacted by applications that require only read-access to the data.

# Oracle and SQL Server Data Type Definitions

The following are the data type definitions referenced in the following database tables.

**Table 7 Oracle and SQL Data Types**

| Descriptive Type | Oracle Type | SQL Server Type | Description |
|---|---|---|---|
| string | varchar2 | nvarchar | Variable length character field. |
| date[a] | date | datetime | Fixed-length data and time field. |
| integer | Number (10) | int | Single-length integer. |
| long | Number (20) | bigint | Double-length integer. |
| text | clob | text | variable-length single-byte character data. |
| float | float | float | Real Number. |
| currency | number | decimal | decimal number with a fixed number of decimal places. |
| variable-length text | clob | ntext | variable-length text up to 1GB (`ntext`) and 4GB (`clob`). |
| variable-length binary | blob | image | variable-length binary file up to 2GB (`image`) and 4GB (`blob`). |

a. For Microsoft SQL Server, the these times are recorded to the nearest millisecond. In the case of an Oracle Server, these times are recorded to the nearest second. This can lead to a slight loss of precision when some values are displayed; for example, metric values from the Business Metrics Schema.

# Business Metrics Schema

This section describes the schema for the business process metrics defined using the Business Process Metrics definer. This is historical data collected as a result of your configuration using the Metrics definer. You can show this data through Business Process Dashboard, or using a reporting application of your choice, for example, Crystal Reports or Microsoft Access.

The Business Metrics schema is based on a dimensional model and the Flow schema is based on an entity-relationship model. This means the Business Metrics schema is designed to maximize the effectiveness of user and application queries. The Flow Schema, which is described in section Flow Schema on page 139, is optimized for application processing and is indexed for this purpose.

Figure 23, Figure 24 and Figure 25 show the dimensional diagrams for the OVBPI business process metrics. In each case, there is a central table, known as the fact table, plus a number of dimension tables. The fact table comprises all the measurements or data that are required for a specific aspect of the metric thresholds. The dimension tables, which are much smaller, comprise descriptive fields. These fields are meaningful when the row headers are presented through user interfaces and reporting applications. The dimension tables are also smaller in order that browse queries on the tables can be completed without delay to the user or application requesting the information.

The metrics data is also used by the Notification Server when generating email alerts relating to threshold violations.

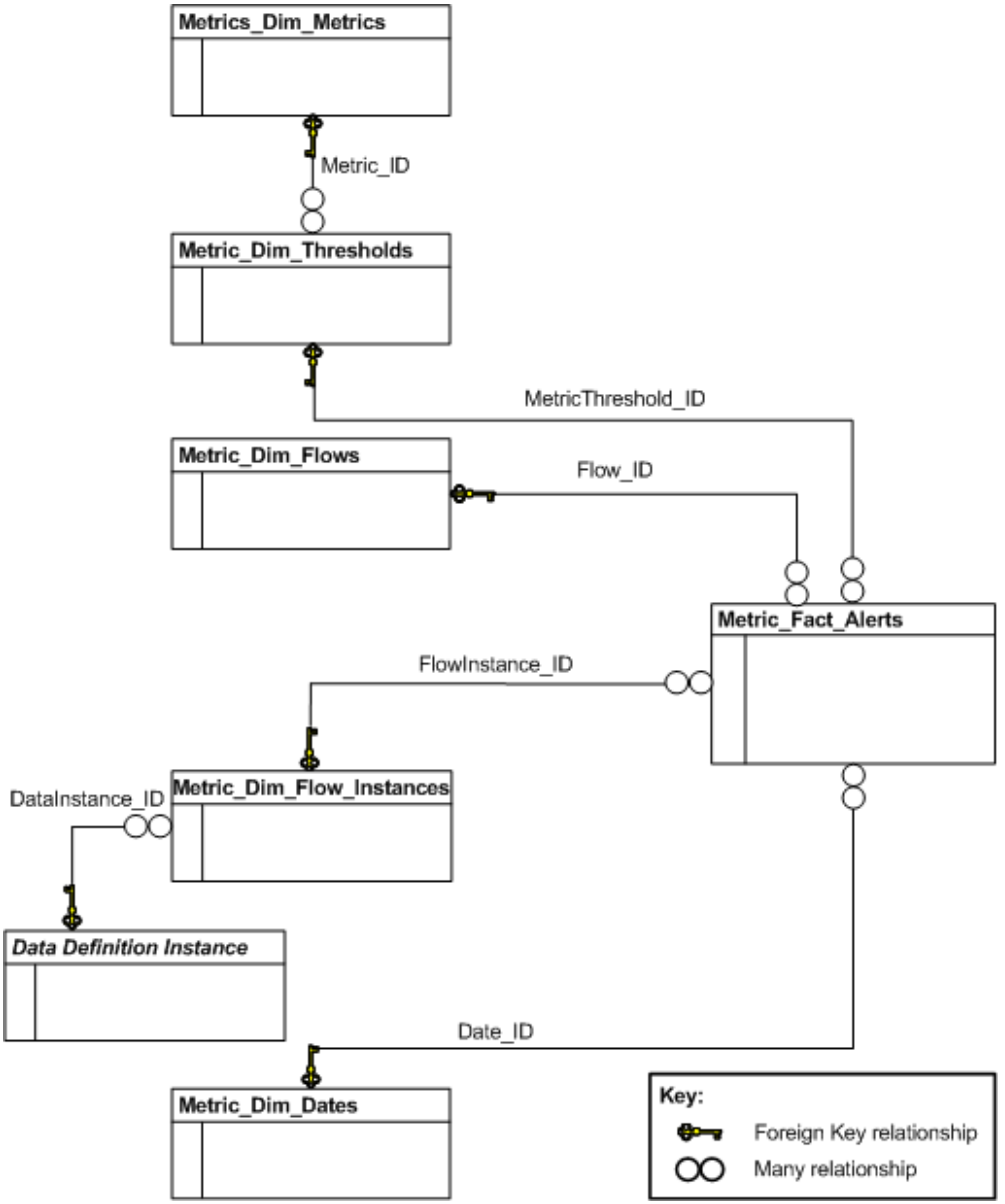
The following sections describe the components that make up the Business Metric schema. The data types listed in the tables are the SQL data types, Table Oracle and SQL Data Types on page 107 describes the SQL Server and Oracle data types mappings and their descriptions.

## Alerts Facts

This dimension provides information relating to the alerts that are generated when a particular threshold that you have defined is violated. The table describing the facts relating to the alerts, contains information for both instance-level alerts and the statistical alerts.

Figure 23 shows the structure of the Metric_Fact_Alerts dimension model and also the dimension tables that are related to it.

**Figure 23 Fact Alerts Dimensional Model**



In Figure 23, the Data Definition Instance table cannot be identified until after the Data definition is deployed, when the table is created and named; refer to section Data Definition Instances on page 159 for details of how to identify this table.

## Business Process Metric Fact Alerts

The Metric_Fact_Alerts table describes the data that is defined for OVBPI alerts that relate to the threshold violations for all business process metrics. The dimension tables shown in Figure 23 are described in section Dimension Tables on page 121.

The primary key is MetricAlert_ID, which is a system-assigned unique identifier.

**Table 8 Metric_Fact_Alerts**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| MetricAlert_ID | string | 36 | No | Primary Key[a]. |
| MetricThreshold_ID | string | 36 | No | Foreign Key[b] to table Metric_Dim_Thresholds. |
| Flow_ID | string | 36 | No | Foreign Key[2] to table Metric_Dim_Flows. |
| Date_ID | string | 36 | No | Foreign Key[2] to table Metric_Dim_Dates. |
| FlowInstance_ID | string | 36 | No | Foreign Key[2] to table Metric_Dim_Flow_Instances. |
| Time | date | | Yes | The time that the threshold alert occurred. |
| RaisedTime | date | | Yes | The time that the threshold alert was raised. |

**Table 8 Metric_Fact_Alerts**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| AlertStatus | string | 12 | Yes | The status of the metric, and can be `Critical`, `Major`, `Minor`, `Warning` and `Normal`. These status correspond to the OpenView Operations severity levels. |
| AlertLevel | integer | | Yes | Numeric equivalent of AlertStatus as follows:<br>• 1 = `Normal`<br>• 2 = `Warning`<br>• 3 = `Minor`<br>• 4 = `Major`<br>• 5 = `Critical` |
| Value | float | | Yes | Metric value or statistic at the time that the alert is generated. The value for this column varies according to the type of threshold. In the case of a Deadline alert, it is the time in seconds since the Deadline was overdue. |
| IsAcknowledged | integer | | Yes | For internal use. |

a. This is the column that uniquely identifies rows.

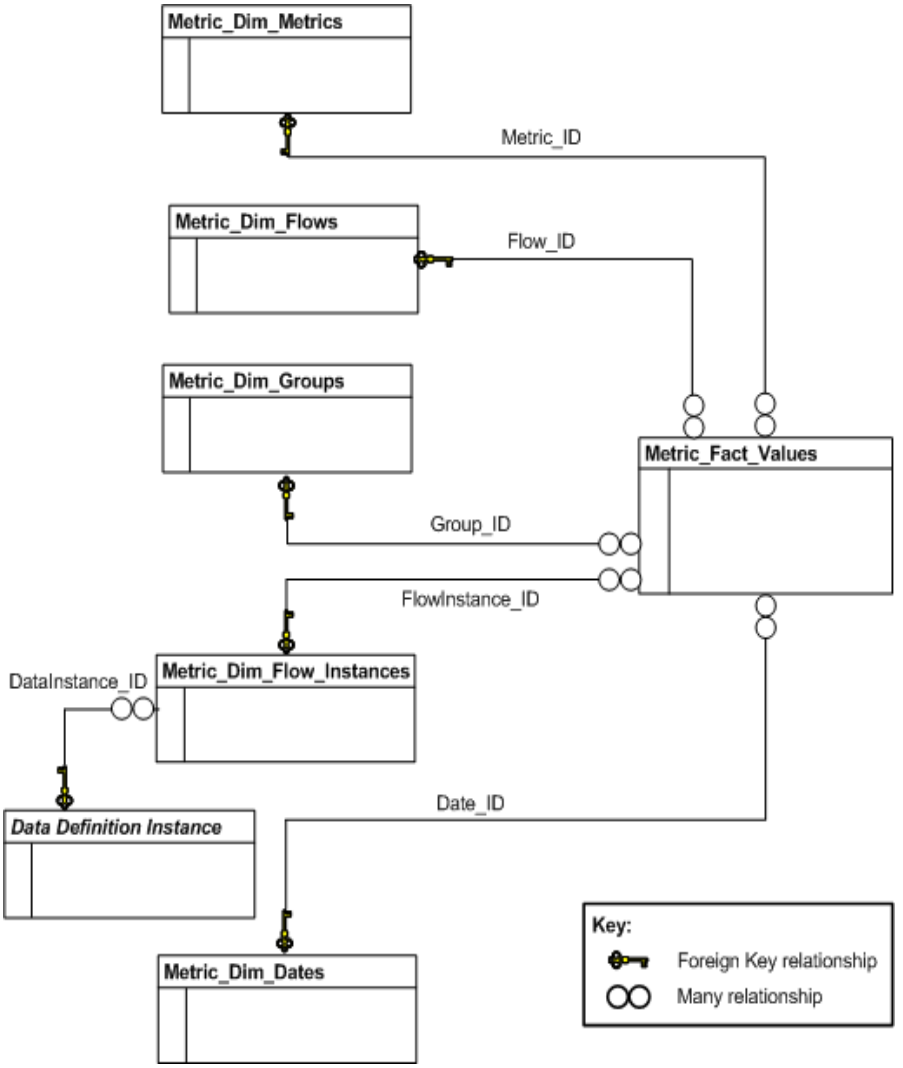
b. This is the column that exists as the primary key in another table.

## Facts Values

This dimension provides information relating to the data collected for the thresholds that you define in the Metrics Definer.

Figure 24 shows the structure of the Fact Values dimension model and also the dimension tables that are related to it.

**Figure 24 Fact Values Dimensional Model**



In Figure 24, the Data Definition Instance table can not be identified until after the Data definition is deployed, when the table is created and named; refer to section Data Definition Instances on page 159 for details of how to identify this table.

## Business Metric Fact Value Dimensions

The Metric_Fact_Values table describes the data that is defined for OVBPI threshold values for the business process metrics that you define. The dimension tables shown in Figure 24 are described in section Dimension Tables on page 121.

The fact values table is linked to the metric name and metric type through the Metric_ID column.

The primary key is MetricValue_ID, which is a system-assigned unique identifier for the date and time.

**Table 9 Metric_Fact_Value**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| MetricValue_ID | string | 36 | No | Primary Key[a]. |
| Metric_ID | string | 36 | No | Foreign Key[b] to table Metric_Dim_Metrics. |
| Flow_ID | string | 36 | No | Foreign Key[2] to table Metric_Dim_Flows. |
| Date_ID | string | 36 | No | Foreign Key[2] to table Metric_Dim_Dates. |
| FlowInstance_ID | string | 36 | No | Foreign Key[2] to table Metric_Dim_Flow_Instances. |

**Table 9 Metric_Fact_Value**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| Status | string | 12 | Yes | The status of the metric, which can be one of:<br>• `Active` - the start condition for the metric has been met, but the end condition has not been met.<br>• `Completed` - the metric has completed and has a value for reporting.<br>• `NoStart` - the metric has completed, but does not have a value.<br>• `Aborted` - the flow has completed before the metric completes. |
| TimeCompleted | date | | Yes | The time (in date format) that the metric completes. |
| Weight | float | | Yes | This column provides the data that you can sort and use to provide a weighting for how important the metric is.<br>This column contains the value of the `Data Definition` property, nominated to be the Weight property, in the OVBPI Modeler. |
| Deadline | date | | Yes | Null, unless the metric has a deadline data item defined, in which case, it contains the value of the deadline. |
| Group_ID | string | 36 | Yes | Foreign Key[2] to table `Metric_Dim_Groups`. If the metric is not part of a group the value is `Null`. |

**Table 9 Metric_Fact_Value**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| Value | float | | Yes | Value of the metric. In the case of the `Single Node` scope metric, the value is equal to the difference between the end time and the start time. If the metric is not complete, that is, there is a start time, but no end time, the value for the metric is `Null`. |
| Idx | integer | | Yes | The order of the Metric conditions, where the conditions are met more than once by a flow instance. The index starts at zero (0) and increments one (1) for every node instance that meets its complete conditions. |
| StartTime | date | | Yes | Time (in date format) when the metric met its start condition. |
| LastUpdateTime | date | | Yes | This column is set each time a change is made to this table for this instance of the metric. It is the time (in date format) that a table row was last updated for the metric instance. |
| DeadlineOverdue | float | | Yes | Applicable only when a Deadline threshold is set for the metric instance. When the metric is complete, this is the number of seconds outside the deadline recorded for the metric instance. This can be a positive or a negative number; a positive value is over the deadline, a negative value is under the deadline. |

a. This is the column that uniquely identifies rows.

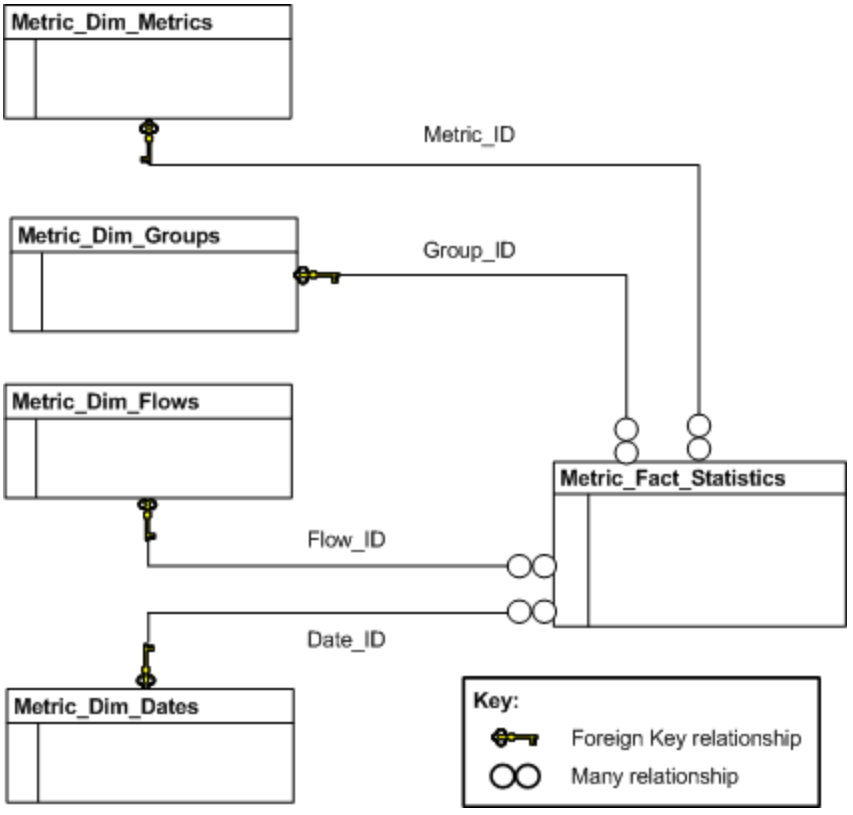
b. This is the column that exists as the primary key in another table.

## Statistics Facts

This dimension provides information relating to the historical data that is recorded for the thresholds that you define in the Metrics Definer.

Figure 25 shows the structure of the Fact Statistics dimension model and also the dimension tables that are related to it.

**Figure 25 Fact Statistics Dimensional Model**



## Business Metric Fact Statistics Dimensions

The Metric_Fact_Statistics table describes the historical data that is defined for OVBPI threshold values related to the metrics that you define. The dimension tables shown in Figure 25 are described in section Dimension Tables on page 121.

There can be multiple rows in this table for a given metric in order to collect statistics on active instances and recently completed instances.

For each set of statistics, data is recorded at time periods that are complete multiples of the value of MeasurementPeriod. As an example, if the measurement period is 10 minutes, the statistics are recorded for the hour, and then in periods of 10 minutes (10, 20 30, 40, 50) after the hour.

If the value of MeasurementPeriod is 24 hours, statistics are recorded at 00:00:00 each day (local time). Be aware that when daylight saving occurs, this period can be 23 hours or 25 hours according to the direction of the change.

The primary key is MetricStatistic_ID, which is a system-assigned unique identifier for the date and time.

**Table 10 Metric_Fact_Statistics**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| MetricStatistic_ID | string | 36 | No | Primary Key[a]. |
| Metric_ID | string | 36 | No | Foreign Key[b] to table Metric_Dim_Metrics. |
| Flow_ID | string | 36 | No | Foreign Key[2] to table Metric_Dim_Flows. |
| Date_ID | string | 36 | Yes | Foreign Key[2] to table Metric_Dim_Dates. |
| Time | date | | Yes | Time that the metric was last updated. |
| IsLatest | integer | | Yes | Indicates whether or not this is the latest statistic. A numeric value of 1 indicates this is the latest statistic. |

**Table 10 Metric_Fact_Statistics**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| StatisticsType | string | 12 | Yes | The type of statistic:<br>• `Active` - all currently active metric instances that relate to this statistic.<br>• `Completed` - all currently completed metric instances that relate to this statistic.<br>• `Total` - the total, since the metric was defined, of all the metrics for the metric instances that have completed. |
| Measurement Period | long | | Yes | The period (in seconds), specified in the Metric Definer, for the metric to be measured over.<br>In the case of total metric instances, this is the period (in seconds) from the first set of metric statistics. |
| Group_ID | string | 36 | Yes | Foreign Key[2] to table `Metric_Dim_Groups`. If the metric is not part of a group, the value is Null. |
| CountTotal | long | | Yes | Number of metric instances to be included in the calculation for the metric statistics; for example, in the case of flow instance metrics this is the number of flow instances. |
| Throughput | float | | Yes | Applies only to completed metrics.<br>The number of metric instances per hour. This is calculated as follows:<br>`(CountTotal * 36,000)/ MeasurementPeriod` |
| SumValue | float | | Yes | Total for the metric instance values. |

**Table 10 Metric_Fact_Statistics**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| SumSquareValue | float | | Yes | Total of the square of the metric instance values. |
| AverageValue | float | | Yes | Average for the metric instance values, |
| StdDevValue | float | | Yes | Standard deviation for the metric instance values. |
| MinimumValue | float | | Yes | The minimum of the metric instance values. |
| MaximumValue | float | | Yes | The maximum of the metric instance values. |
| WeightTotal | float | | Yes | Total of the Flow instance Weight attribute for the metric; for example, in the case of an Order Flow instance, this might be the total value of the orders (a historical total). |
| Weight Throughput | float | | Yes | This applies only to Completed metrics.<br>This is number of instance per hour that relate to this metric. The value is calculated as follows:<br>`(WeightTotal * 36,000) / MeasuremenPeriod` |
| WeightSumValue | float | | Yes | Total of metric instance values * Flow instance weight |
| WeightSum SquareValue | float | | Yes | Total of the square of the metric instance values * Flow instance weight |

**Table 10 Metric_Fact_Statistics**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| WeightAverage Value | float | | Yes | Average of metric instance values weighted by flow instances that have a higher weight than the average. |
| WeightStdDev Value | float | | Yes | Standard deviation for the metric instance values weighted by flow instances that have a higher weight than the average. |

a. This is the column that uniquely identifies rows.

b. This is the column that exists as the primary key in another table.

# Dimension Tables

The following tables describe the business dimensions defined for the fact tables described above.

## Business Metrics Dimensions

The Metric_Dim_Metrics table is the dimensional table that you can use for creating reports from business process metric information collected by the Business Impact Engine. This is a dimension on the Metric_Fact_Alerts table, Metric_Fact_Statistics table and Metric_Fact_value table.

The primary key is Metric_ID, which is a system-assigned unique identifier.

Flow_ID is a foreign key used to link to the Flow table, which identifies the flow definition for the business process metric.

**Table 11 Metric_Dim_Metrics**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| Metric_ID | string | 36 | No | Primary Key[a]. |
| MetricName | string | 120 | No | Name given to a business process metric through the Business Process Metric definer. |
| Flow_ID | string | 36 | No | Foreign Key[b] to the Flows table; see Table 20 on page 140. |
| ValueUnits | string | 12 | Yes | Specifies the units for the business process metric type; for example, seconds. |
| CreatedDate | date | | Yes | Time and date when the metric is created. |
| Measurement Period | long | | Yes | The period, in units of seconds, that the business process metric. |
| GroupName | string | 120 | Yes | Name used for grouping statistics. |

**Table 11 Metric_Dim_Metrics**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| LastStatistics RecordTime | date | | Yes | Time and date for the last statistics report. This is used to work out when to start recording historical information. |
| IsDeleted | integer | | Yes | Indicates whether or not a metric has been deleted. A setting of one (1) means the metric is deleted. |

a. This is the column that uniquely identifies rows.

b. This is the column that exists as the primary key in another table.

## Business Metrics Date Dimensions

The Metric_Dim_Dates table is the dimensional table that you can use for creating reports that contain time dimensions for the Metric_Fact_Alerts table, Metric_Fact_Statistics table and Metric_Fact_value table.

The primary key is Date_ID, which is a system-assigned unique identifier for the date and time.

**Table 12 Metric_Dim_Dates**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| Date_ID | string | 36 | No | Primary Key[a] |
| Year | integer | | No | The `year` for the time, for example, 2005. |
| Quarter | integer | | No | A number representing a three month period for each quarter of a year as follows:<br>• 1 = January to March<br>• 2 = April to June<br>• 3 = July to September<br>• 4 = October to December |

**Table 12 Metric_Dim_Dates**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| Month | string | 30 | No | The month for the time, specified as a string, in the configured locale for the database. |
| Month_Num | integer | | No | The `month` for the time, represented as a numeral between 1 and 12. |
| Day | integer | | No | The `day` for the time, represented as a numeral between 1 and 31. |
| Week | integer | | No | The week in the year for the time, represented as a numeral between 1 and 53. |
| DayOfWeek | string | 30 | No | The day of the week, specified as a string, in the configured locale for the database. |
| DayOfWeek_Num | integer | | No | The day of the week for the time, represented as a numeral between 1 and 7 as follows:<br>• 1 = Sunday<br>• 2 = Monday<br>• 3 = Tuesday<br>• 4 = Wednesday<br>• 5 = Thursday<br>• 6 = Friday<br>• 7 = Saturday |
| Hour | integer | | No | The `hour` for the time, represented as a numeral between 0 and 23. |

**Table 12 Metric_Dim_Dates**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| Minute | integer | | No | The `minute` for the time, represented as a numeral between 0 and 59. |
| Time | date | | No | The date and time, rounded down to the nearest minute, in date format as supported by your database. |

a. This is the column that uniquely identifies rows.

## Business Metric Flows Dimensions

The Metric_Dim_Flows table is the dimensional table that represents the flows that you want to report thresholds for. This is a dimension on the Metric_Fact_Alerts table, Metric_Fact_Statistics table and Metric_Fact_value table.

The primary key is Flow_ID, which is a system-assigned unique identifier.

**Table 13 Metric_Dim_Flows**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| Flow_ID | string | 36 | No | Primary Key[a]. |
| FlowName | string | 120 | No | The name of the business flow as entered in the OVBPI Modeler. |

**Table 13 Metric_Dim_Flows**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| Status | string | 12 | Yes | The current status of this flow, which can be one of `Active`, `Impeded`, `Blocked` or `Deleted`, where:<br>• `Active` means the flow has been deployed.<br>• `Impeded` means that at least one instance of the flow is in the `Impeded` state; see `Status` in Table 21 on page 142.<br>• `Blocked` means that at least one instance of the flow is in the `Blocked` state; see `Status` in Table 21 on page 142.<br>• `Deleted` means the flow has been undeployed. |
| DataDefinition_ID | string | 36 | Yes | Primary entity identifier for the Data definition associated with the Flow. |

a. This is the column that uniquely identifies rows.

## Business Metrics Flows Instance Dimensions

The Metric_Dim_Flow_Instances table is the dimensional table that represents the flow instances that you want to report thresholds for. This is a dimension on the Metric_Fact_Alerts table and Metric_Fact_value table.

The primary key is FlowInstance_ID, which is a system-assigned unique identifier for the date and time.

**Table 14 Metric_Dim_Flow_Instances**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| FlowInstance_ID | string | 36 | No | Primary Key[a]. |
| Flow_ID | string | 36 | No | Foreign Key[b] to table Flows (Flow_ID) column. |
| Identifier | string | 120 | Yes | The identity of the related Data definition for this Flow.<br>When you create a flow using the OVBPI Modeler, you can enter the name of a property of a related Data Definition to be used as an Identity property. This column holds the value of the Identity property for the specific flow instance. |
| DataDefinition_ID | string | 36 | Yes | A unique identifier (GUID) for the business object class associated with this metric instance; this column is taken from `Model_ID` in the Data Object table. An example of a `Model_ID` is an Order Definition. |
| DataInstance_ID | string | 36 | Yes | A unique identifier (GUID) for the business object instance associated with this metric instance |
| Weight | float | | Yes | The value of the Weight for the flow instance. |

**Table 14 Metric_Dim_Flow_Instances**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| StartTime | date | | Yes | The time that this flow instance was last started (in date format). |
| EndTime | date | | Yes | The time that this flow instance was last completed (in date format). |
| Status | string | 12 | Yes | The current status of the flow instance for this metric, which can be one of `Active` or `Completed`, where:<br>• `Active` means the flow instance has met the start condition for at least one node in the flow.<br>• `Completed` means the flow instance has met the complete conditions for one of the end nodes in the flow.<br>The Metric Engine does not report on the interim status of flow instances (impeded or blocked). For the purpose of the metrics data, all flow instances are active until they have completed. |

a. This is the column that uniquely identifies rows.

b. This is the column that exists as the primary key in another table.

## Business Metric Group Dimensions

The Metric_Dim_Groups table is the dimensional table that represents the groups that you might have defined for your metrics in the Metrics Definer. This is a dimension on the Metric_Fact_Statistics table and Metric_Fact_value table.

The primary key is Group_ID, which is a system-assigned unique identifier.

**Table 15 Metric_Dim_Groups**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| Group_ID | string | 36 | No | Primary Key[a]. |
| GroupName | string | 120 | No | The name of the group that this metric belongs to; for example, you might have a group called Terminals that represents airport terminals. |
| GroupValue | string | 256 | Yes | The values of the groups identified in GroupName; for example, in the case of Terminals, you might have the values 1, 2 and 3, for Terminals 1, 2 and 3. |

a. This is the column that uniquely identifies rows.

## Business Metric Threshold Dimensions

The Metric_Dim_Thresholds table is the dimensional table that represents the thresholds defined for your metrics. When a threshold for an instance is violated, an alert is generated for each instance violated. When a threshold for a statistic is violated, an alert is generated each time the violation level changes for the statistic.

This is a dimension on the Metric_Fact_Alerts table.

The primary key is MetricThreshold_ID, which is a system-assigned unique identifier.

**Table 16 Metric_Dim_Thresholds**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| MetricThreshold_ ID | string | 36 | No | Primary Key[a]. |
| Metric_ID | string | 36 | No | Foreign Key[b] to table Metric_Dim_Metrics. |
| ThresholdName | string | 120 | No | Name of the threshold; used for email notifications to distinguish between alerts. |
| Threshold Description | string | 256 | Yes | Description of the threshold. |
| ThresholdMessage | string | 256 | Yes | A message that can be included in an email notification to provide additional information and context about the alert for the recipient. |

**Table 16 Metric_Dim_Thresholds**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| ThresholdType | string | 12 | No | Indicates whether or not the threshold is set for a particular instance, or for a particular statistics type as follows:<br>• `Instance` - set for a particular instance<br>• `Active` - set for active instances<br>• `Completed` - set for instances that have completed during the metric period<br>• `Total` - set for all instances that have completed since the metric was defined. |
| ThresholdColumn Name | string | 40 | Yes | Defines the column that the metric is set for as follows:<br>• `CountTotal` - number of instances<br>• `Throughput` - throughput<br>• `AverageValue` - average<br>• `MinimumValue` - minimum value<br>• `MaximumValue` - maximum value<br>• `Null`- when `ThresholdType` is `Instance` from `Value` column in `Metric_Fact_Value` table.<br>• `WeightTotal` - weight total<br>• `WeightThroughput` - weight throughput<br>• `WeightAverageValue` - weight average |

**Table 16 Metric_Dim_Thresholds**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| ThresholdTest | string | 12 | No | Defines the type of test applied and the meaning of the alert level thresholds. With the exception of Deadline, these are taken from the `Value` columns of the metric_fact_values, or the SumValue column of the metric_fact_statistics tables:<br>• `OverValue` - value is equal to or more than value specified in the alert level.<br>• `UnderValue` - value is less than or equal to value specified in the alert level.<br>• `OverUsual` - number of standard deviations that the value is equal to or above the average for this metric (applicable to instance, average and weighted average only)<br>• `UnderUsual` - number of standard deviations that the value is equal to or below the average for this metric (applicable to instance, average and weighted average only)<br>• `Unusual` - the value of the metric is either above or below the average for this metric, that is, either the `OverUsual` or `UnderUsual` conditions are met.<br>• `Deadline` - the metric has completed within the required time. This is applicable only to instance thresholds. This is the value in the `Deadline` column in the metric_fact_values table. |

**Table 16 Metric_Dim_Thresholds**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| AlertCriticalLevel | float | | Yes | Critical threshold level |
| AlertMajorLevel | float | | Yes | Major threshold level |
| AlertMinorLevel | float | | Yes | Minor threshold level |
| AlertWarning Level | float | | Yes | Warning threshold level |
| LastCheckTime | date | | Yes | Last time that the threshold was checked by the Metrics Engine. |
| CurrentAlert Status | string | 12 | Yes | The highest level recorded for the alert status of a metric instance within the current Collection interval, and can be `Critical`, `Major`, `Minor`, `Warning` and `Normal`. These status correspond to the OpenView Operations severity levels. |
| CurrentAlertLevel | integer | | Yes | Numeric equivalent of CurrentAlertStatus as follows:<br>• 1 = `Normal`<br>• 2 = `Warning`<br>• 3 = `Minor`<br>• 4 = `Major`<br>• 5 = `Critical` |
| CurrentAlert FromTime | date | | Yes | The start date and time of the start of the current Collection interval for the CurrentAlertLevel and CurrentAlertStatus values. If the Collection interval is zero (not set), then this is the data and time of the start of the current threshold polling interval. |

**Table 16 Metric_Dim_Thresholds**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| StagingAlertLevel | integer | | Yes | The highest level recorded for the alert status of a metric instance since the end of the current Collection interval. Used internally as an ongoing record of the metric instance alert status in readiness for the next Collection interval. |
| AlertCriticalLevel DefnUnit | string | 12 | Yes | The unit used to define the Critical Alert value for the Metric Threshold within the Metric Definer. |
| AlertMajorLevel DefnUnit | string | 12 | Yes | The unit used to define the Major Alert value for the Metric Threshold within the Metric Definer. |
| AlertMinorLevel DefnUnit | string | 12 | Yes | The unit used to define the Minor Alert value for the Metric Threshold within the Metric Definer. |
| AlertWarning LevelDefnUnit | string | 12 | Yes | The unit used to define the Warning Alert value for the Metric Threshold within the Metric Definer. |
| CreatedDate | date | | Yes | Time and date when the threshold is created. |

a. This is the column that uniquely identifies rows.

b. This is the column that exists as the primary key in another table.

Be aware that the `ThresholdTest` value `Deadline` has the following behavior:

- In the case of metrics for Node instances, an alert is generated if the node is not completed for cases where the node is not started.
- The alert levels are the number of seconds before the deadline occurs, for example, it is possible to have a warning alert (`AlertWarningLevel`) at four hours before the deadline and a critical alert (`AlertCriticalLevel`) when the deadline is reached.

## Business Metric Custom Types

The Metrics Definer enables you to define your own custom metrics.

The Metric_CustomTypes table is the dimensional table that represents the custom metrics that you can define and are described in Table 17 on page 134. The Metrics Definer uses this table when presenting the custom metrics that you can select.

The primary key is CustomMetricName, which is the unique name that is given to the custom metric when you define it.

**Table 17 Metric_CustomTypes**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| CustomMetric Name | string | 120 | No | Primary Key[a].<br>This is the name that appears in the Metric definer as the `Metric value type` identifier. |
| CustomMetric Description | string | 256 | Yes | Description for the custom metric. |
| CustomSPName | string | 120 | No | Name of the stored procedure associated with this custom metric. |
| ValueUnits | string | 12 | Yes | Specifies the units for the custom-metric type. This string can be shown through your presentation interfaces (for example a reporting tool that you use). |

a. This is the column that uniquely identifies rows.

Defining custom metrics is described in more detail in the *HP OpenView Business Process Insight Integration Training Guide - Modeling Flows*.

## Business Metric Failure Messages

All SQL errors generated by the custom metrics that you define are propagated to the Business Impact Engine, which logs an errors directly into its log file.

Always refer to the Business Impact Engine log file when you want to identify errors with the custom metrics that you define.

## Metric Views

There are two additional database tables that are defined for compatibility with the metrics tables provided in previous versions of OVBPI. These are:

- Metrics view (Table 18 on page 135)
- Metrics_Value view (Table 19 on page 137)

Table 18 on page 135 is a view onto the Metrics_Dim_Metrics table described in section Business Metrics Dimensions on page 121.

**Table 18 Metrics**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| MetricName | string | 40 | No | Primary Key[a] and the name given to the metric through the OVBPI Modeler. |
| MetricType | string | 6 | Yes | The type of metric, for example:<br>• NET (Single Node scope)<br>• TBN (Multiple Node scope)<br>• FET (Whole Flow scope)<br>• CUSTOM (custom-written metric) |
| Flow_ID | string | 36 | No | Primary Key[1] and Foreign Key[b] to the Flows table; Table 21 on page 142. |

**Table 18 Metrics**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| StartNode | string | 36 | Yes | In the case of NET and TBN, this is a unique identifier of the node where the metric data collection is to be started.<br>In the case of FET, `StartNode` is null.<br>In the case of CUSTOM, `StartNode` is either a unique identifier, or null, depending on whether it is for one or more nodes within the flow (unique identifier), or a whole flow (null). |
| StartCondition | integer | | Yes | For internal use. |
| EndNode | string | 36 | Yes | In the case of NET and TBN, this is a unique identifier of the node where the metric data collection is to be stopped.<br>In the case of FET, `StartNode` is null.<br>In the case of CUSTOM, `EndNode` is either a unique identifier, or null, depending on whether it is for one or more nodes within the flow (unique identifier), or a whole flow (null). |
| EndCondition | integer | | Yes | For internal use. |

a. This is the column that uniquely identifies rows.

b. This is the column that exists as the primary key in another table.

## Metric Values

The Metric_Values view holds the details of the metrics data for flow instances. There is an entry for each flow instance for each metric in the flow.

**Table 19 Metric_Values**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| Name | string | 40 | No | Metric name. |
| Type | string | 6 | Yes | The type of metric, for example:<br>• NET (Single Node scope)<br>• TBN (Multiple Node scope)<br>• FET (Whole Flow scope)<br>• CUSTOM (custom-written metric) |
| Flow_Instance | string | 36 | Yes | Foreign Key[a] to Flow_Instance table; see Table 21 on page 142. |
| StartTime | date | | Yes | Time when flow instance met the Start condition for `StartNode` (in date format); see Table 18 on page 135 for a description of `StartNode`. |
| StartTimeLong Millis | long | | Yes | Time (in milliseconds) when flow instance met the Start condition for `StartNode`; see Table 18 on page 135. |
| EndTime | date | | Yes | Time when flow instance met the complete condition for `EndNode` (in date format); see Table 18 on page 135. |
| EndTimeLong Millis | long | | Yes | Time (in milliseconds) when flow instance met the complete condition for `EndNode`; see Table 18 on page 135. |
| Value | float | | Yes | For internal use. |

**Table 19 Metric_Values**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| Time | date | | Yes | The time that the metric was updated. |
| Idx | integer | | Yes | Index or counter for an instance where the metric conditions are met multiple times by one flow instance, for example, when the instance is in a loop condition within the flow. |

a. This is the column that exists as the primary key in another table

# Flow Schema

This section describes the schema for the business flows defined using the OVBPI Modeler; this data is used by the Business Impact Engine to process the impact data collected from the flows that you define.

If you want to report measurements for your flows, use the business process metrics data held in the metrics schema as described in section Business Metrics Schema on page 108. These business process metric tables hold all the measurement data that you have requested is to be recorded and are designed to be effective for queries and searches. The Flow schema is an entity-relationship schema and is designed for transaction processing. The Flow tables hold all the data collected for the Flows and Flow instances that you have defined and deployed using the OVBPI Modeler. Use this information to show details of the Flow through the Dashboard.

If you have configured an Oracle Server to store your Flow Schema data, the length of some of the fields in the Flow Schema tables is three-times the length shown. The fields affected are user defined labels, such as names. Fields that are internally generated, such as identifiers are not impacted. This allows for the fact that UTF8 can be used as the database character set, in which case, up to three bytes might be required to store each character.

As an example, the length of the `FlowName` column is 120 rather than 40.

As the flow data stored in the database represents a dynamic data model, the schema is designed with performance as the design goal, rather than reporting techniques. To improve performance, database indexes, which improve the rate of access to the database tables, are defined for both Oracle and SQL Server.

The database objects for the Flow schema are shown as an entity-relationship diagram; see Figure Flow Schema Entity-Relationship Diagram on page 156.

The following sections describe the components that make up the flow schema.

The data types listed in the tables are the SQL data type. Table Oracle and SQL Data Types on page 107 describes the SQL Server and Oracle data types mappings and their descriptions.

## Flows

The Flows table is the primary table as it details the identities of the Flow definitions. As is shown in Flow Schema Entity-Relationship Diagram on page 156, The Flow table is related to the Nodes, Arcs and Flow_Instances tables. The primary key for Flows is the Flow_ID, a unique identifier derived from the Flow definition's Java object model.

**Table 20 Flows**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| Flow_ID | string | 36 | No | Primary Key[a]. |
| FlowName | string | 120 | No | The name of the business flow as entered in the OVBPI Modeler. |
| FlowDescription | string | 256 | Yes | The description of the business flow as entered in the OVBPI Modeler. |
| AvrgTime | float | | Yes | The average time taken to complete all flow instances. |
| SampleCount | integer | | Yes | Internal value used for calculating the average time, |
| ActiveFlows | integer | | Yes | The number of flow instances currently being monitored. |
| TotalFlows | integer | | Yes | The total number of flow instances, including the flow instances that have completed. |

**Table 20 Flows**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| Status | string | 12 | Yes | The current status of this flow, which can be one of `Active, Impeded`, `Blocked` or `Deleted`, where:<br>• `Active` means the flow has been deployed.<br>• `Impeded` means that at least one instance of the flow is in the `Impeded` state; see `Status` in Table 21 on page 142.<br>• `Blocked` means that at least one instance of the flow is in the `Blocked` state; see `Status` in Table 21 on page 142.<br>• `Deleted` means the flow has been undeployed. |
| RepositoryId | string | 36 | Yes | A unique identifier that can be used to associate different versions of the same object; different versions of a Flow definition or Data definition have different Flow_IDs, but they will have the same RepositoryId. |
| Subclass | string | 256 | Yes | For internal use. |
| Primary_entity | string | 36 | Yes | A unique identifier (GUID) for the business object class associated with this Flow definition; this column is taken from `Model_ID` in the Data Object table. An example of a `Model_ID` is an Order Definition. Identifies the Data definition for the Flow. |

**Table 20 Flows**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| Repository Revision | integer | | Yes | The revision number identifying the the version of the Flow that is held in the Repository table. |
| FolderPath | string | 604 | Yes | For internal use. |

a. This is the column that uniquely identifies rows.

## Flow Instance

The Flow_Instance table describes each instance of a specific flow. The primary key is FlowInstance_ID, which is a system-assigned unique identifier.

Primary_entity and Primary_entity_inst are two foreign keys that can be used to link the Flow schema to the Business Entity schema.

**Table 21 Flow_Instance**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| FlowInstance_ID | string | 36 | No | Primary Key[a]. |
| Flow_ID | string | 36 | No | Foreign Key[b] to table Flows (`Flow_ID`) column. |
| Identifier | string | 120 | Yes | The identity of the related Data definition for this Flow.<br>When you create a flow using the OVBPI Modeler, you can enter the name of a property of a related Data Definition to be used as an Identity property. This column holds the value of the Identity property for the specific flow instance. |

**Table 21 Flow_Instance**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| Primary_entity | string | 36 | Yes | A unique identifier (GUID) for the business object class associated with this flow instance; this column is taken from `Model_ID` in the Data Object table. An example of a `Model_ID` is an Order Definition. Identifies the Data definition for the Flow. |
| Primary_entity_inst | string | 36 | Yes | A unique identifier (GUID) for the business object instance associated with this flow instance; this column is taken from `InstanceTable` in the Data Object table. Identifies the instance of the Data Definition for the Flow instance. |
| Weight | float | | Yes | This column provides the data that you can sort and use to provide a weighting for how important the instance is.<br>This column contains the value of the Data Definition property, nominated to be the Weight property, in the OVBPI Modeler.<br>You can use this column to use to sort flow instances into a required order, for example, it might be the value of an order expressed as a classification, such as Gold/Silver/Bronze.<br>This information is mapped from the instance of the Data definition and provides a quick view of the data object from within the flow instance. |

**Table 21 Flow_Instance**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| Weight_type | string | 120 | Yes | This column contains the name of the Data definition property nominated to be the `Weight` property for this flow in the OVBPI Modeler.<br>The `Weight_type` column provides the context for the values in the Weight column.<br>As an example, the Business Process Dashboard might have an option to view Orders sorted by Order Value, where the actual value of the order ($25) is the Weight and Order Value is the Weight type. |
| StartTime | date | | Yes | The time that this flow instance was last started (in date format). |
| StartTimeLong Millis | long | | Yes | The time (in milliseconds) that the flow instance was last started. |
| EndTime | date | | Yes | The time that this flow instance was last completed (in date format). |

**Table 21 Flow_Instance**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| EndTimeLong Millis | long | | Yes | The time (in milliseconds) that the flow instance was last completed. |
| Status | string | 12 | Yes | The current status of this flow instance, which can be one of `Active`, `Impeded`, `Blocked` or `Completed`, where:<br>• `Active` means the flow instance has met the start condition for at least one node in the flow.<br>• `Impeded` means that the flow instance cannot progress to completion without encountering a blocked node.<br>• `Blocked` means that the flow instance is active at a node where the node's services have failed.<br>• `Completed` means the flow instance has met the complete conditions for one of the end nodes in the flow. |
| Subclass | string | 256 | Yes | For internal use |

a. This is the column that uniquely identifies rows.

b. This is the column that exists as the primary key in another table.

## Nodes

The Nodes table details individual nodes defined for a flow. The nodes are not defined as reusable entities as this property is only useful during the modeling stage. Representing the node as a unique entity also makes is easier to create queries and views. The primary key for the Nodes table is Node_ID and the foreign key is Flow_ID, which relates the nodes to their parent flow.

**Table 22 Nodes**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| Node_ID | string | 36 | No | Primary Key[a]. |
| Flow_ID | string | 36 | No | Foreign Key[b] to the table Flows (`Flow_ID`). |
| NodeName | string | 120 | No | The name of the node, taken from the OVBPI Modeler. |
| NodeDescription | string | 256 | Yes | The description of the node, taken from the OVBPI Modeler. |
| NodeType | string | 12 | Yes | The node type, which can be one of `START`, `WORK`, `END`. |
| X_Pos | integer | | Yes | The X coordinate of the node icon position on the flow graph for display purposes. |
| Y_Pos | integer | | Yes | The Y coordinate of the node icon position on the flow graph for display purposes. |
| ActiveCount | integer | | Yes | The numbers of current, or active, flow instances at this node. |
| TotalCount | integer | | Yes | The number of times the node has been started. |

**Table 22 Nodes**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| TotalTime | float | | Yes | The accumulated time that all the node instances for this node have been active (or the accumulated time for each progression of the node). |
| AvrgTime | float | | Yes | The average time taken to complete this node instance. |
| SampleCount | integer | | Yes | Internal value used for calulating the average time, |
| ActiveWeight | float | | Yes | Total value of weight parameter for flow instances that are currently active at this node. |
| TotalWeight | float | | Yes | Total value of weight parameter for all flow instances that have been processed at this node. |
| InstanceRate | float | | Yes | The measure of the number of instances per hour that are currently being processed at this node. |
| WeightRate | float | | Yes | The measure of the weight per hour for the flow instances that are currently being processed at this node, for example, the value of customer orders per hour currently being processed at this node. |

**Table 22 Nodes**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| LastRateUpdate | date | | Yes | The time that the weight and instance rates were last updated (in date format). |
| LastRateUpdate LongMillis | long | | Yes | The time (in milliseconds) that the weight and instance rates were last updated. |
| ResourceStatus | string | 12 | Yes | The status of the service, and can be `Critical`, `Major`, `Minor`, `Warning` and `Normal`. These status correspond to the OpenView Operations severity levels. If the node uses more than one service, this field is set to the most serious status for the services used. |

a. This is the column that uniquely identifies rows.

b. This is the column that exists as the primary key in another table.

## Node Instance

The Node_Instance table details the instances of nodes associated with each of the flow instances. The primary key for this table is NodeInstance_ID.

**Table 23 Node_Instance**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| NodeInstance_ID | string | 36 | No | Primary Key[a]. |
| FlowInstance_ID | string | 36 | No | Foreign Key[b] to table Flow_Instance (`FlowInstance_ID`). |
| Node_ID | string | 36 | No | Foreign Key[2] to table Nodes (`Node_ID`). |

**Table 23 Node_Instance**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| StartTime | date | | Yes | The time the node was last started (in date format). |
| StartTimeLong Millis | long | | Yes | The time (in milliseconds) that the node was last started. |
| EndTime | date | | Yes | The time the node last completed (in date format). |
| EndTimeLong Millis | long | | Yes | The time (in milliseconds) that the node was last completed. |
| Status[c] | string | 16 | No | The status of the node instance as represented by the most recent event and can be one of `Initial`, `Started`, `Started Again` and `Completed`. |
| ResourceStatus | string | 12 | Yes | The status of the service, and can be `Critical`, `Major`, `Minor`, `Warning` and `Normal`. These status correspond to the OpenView Operations severity levels. If the node uses more than one service, this field is set to the most serious status for the services used. |

a. This is the column that uniquely identifies rows.

b. This is the column that exists as the primary key in another table.

c. `Initial` indicates that neither the start condition nor the complete condition for the node have been met. `Started` means the start condition for the node have been met. `Started Again` means the start condition for a node have been met more than once, but does not indicate that any complete conditions have been met. `Completed` means that the complete conditions for the node have been met, but does not indicate that any start conditions have been met. The Business Impact Engine evaluates the status of a node instance based on the order in which the events were submitted, not by the order in which they were received.

## Node Instance Started Times

The Node_Instance_StartedTimes table lists the times that a node instance is started, as a node instance can be started multiple times. This table is linked to the Node_Instance table though the NodeInstance_ID column. As it is possible to progress through a node more than once, a single node can have multiple database entries. The Idx column distinguishes each of these entries in the database.

**Table 24 Node_Instance_StartedTimes**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| NodeInstance_ID | string | 36 | No | This column, plus Idx make up the Primary Key. |
| Idx | integer | | Yes | The order of the start times.The index starts at zero (0) and increments one (1) for every time a node instance that meets it start condition. |
| StartedTime | date | | Yes | The time that the start condition for this node instance were met (in date format). |
| StartedTimeLong Millis | long | | Yes | The time (in milliseconds) that the start conditions for this node instance were met. |

## Node Instance Completed Times

The Node_Instance_CompletedTimes table lists the times that the node instance meets its complete conditions, as a node instance can be completed multiple times. It is linked to the Node_Instance table through the NodeInstance_ID column.As it is possible to progress through a node more than once, a single node can have multiple database entries. The Idx column distinguishes each of these entries in the database.

**Table 25 Node_Instance_CompletedTimes**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| NodeInstance_ID | string | 36 | No | This column, plus `Idx` make up the Primary Key. |
| Idx | integer | | Yes | The order of the completed times.The index starts at zero (0) and increments one (1) for every time a node instance that meets it complete condition. |
| CompletedTime | date | | Yes | The time that the complete conditions for this node instance is met In date format). |
| CompletedTime LongMillis | long | | Yes | The time (in milliseconds) that the complete conditions for this node instance were met. |

## Arcs

Arcs are a component of the Flow definition and describe the links between the nodes that create a particular flow.

Arcs have a many-to-one relationship with the Flows table.

**Table 26 Arcs**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| Flow_ID | string | 36 | No | Foreign Key[a] to table Flows (`Flow_ID`). |
| Source | string | 36 | No | GUID of source node Node_ID. |
| Destination | string | 36 | No | GUID of destination node Node_ID. |
| Type | string | 12 | Yes | Type can be `NORMAL` or `SEQUENCE`. `SEQUENCE` indicates a Check Sequence arc is defined. |

a. This is the column that exists as the primary key in another table.

## Services

The Resources table represents the services, or the IT applications and utilities that a node is linked to and is required to monitor, for example, the Take Order node might rely on a Web site and e-commerce application. These are the services that have been defined in OVIS and OpenView Operations

and that have been linked into a specific business flow.

**Table 27 Resources**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| Resource_ID | string | 296 | No | Primary Key[a]. |
| ResourceName | string | 256 | No | The name of the service. |
| ResourceType | string | 40 | No | Identifies the type of service and can be one of `OVIS`, `OVSN`, `SALS` or `SOAM` `OVSN` is used to indicate either an OpenView Service Navigator and OpenView Operations for Windows Service. `SALS` is used to indicate a Standalone Service. `OVIS` and `SOAM` are used to represent OpenView Internet Services and SOA Manager Services respectively. |
| Resource Description | string | 256 | Yes | A description of the service as entered in the OVBPI Modeler. |
| Status | string | 12 | Yes | The status of the service, and can be `Critical`, `Major`, `Minor`, `Warning` and `Normal`. These status correspond to the OpenView Operations severity levels. |
| RootCause | text | | Yes | The current root cause obtained from OVIS or OpenView Operations. |

**Table 27 Resources**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| LastChange | date | | Yes | The time that the service status was last modified (in date format). |
| LastChangeLong Millis | long | | Yes | The time (in milliseconds) that the service status was last modified. |
| Deployment Status | string | 12 | Yes | The current deployment status of this service, which can be one of `Active` or `Deleted`, where:<br>• `Active` means the service is deployed.<br>• `Deleted` means the service is undeployed, but is still used by a flow. |

a. This is the column that uniquely identifies rows.

## Node2Resources

The Node2Resources table maps nodes to services.

As services are re-usable, it is possible for nodes and services to have a many-to-many relationship, that is, many services can be used by many nodes and more than one node can use more than one service. As an example, the Get Order and Update Order nodes might both link to the Web site and e-commerce services.

**Table 28 Node2Resources**

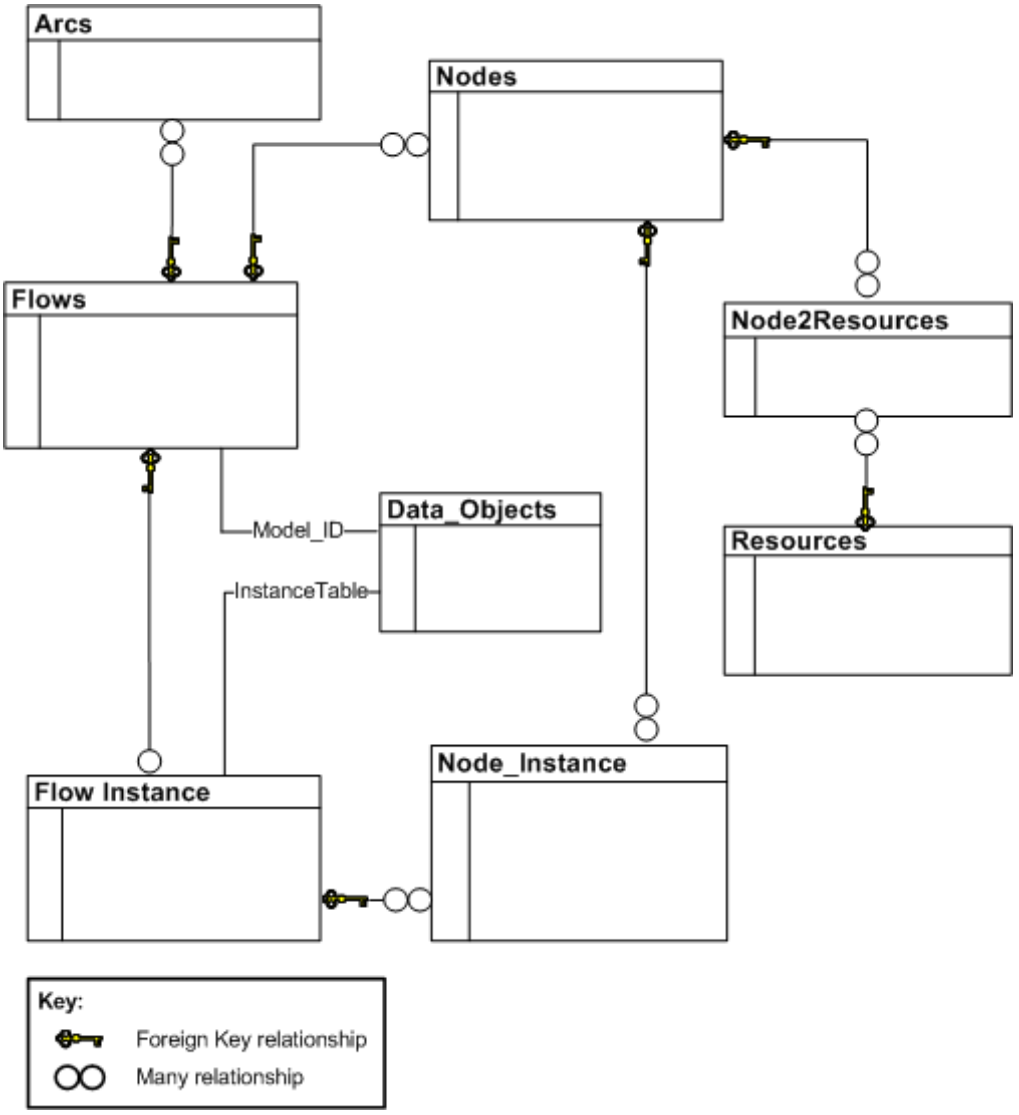
| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| Node_id | string | 36 | No | Foreign Key[a] to table Nodes (`Node_ID`). |
| Resource_ID | string | 296 | No | Foreign Key[1] table Services (`Resource_ID`). |

a. This is the column that exists as the primary key in another table.

# Entity Relationships for Flow Schema

The following diagram shows the relationship between the database tables for the Flow schema.

**Figure 26 Flow Schema Entity-Relationship Diagram**



# Business Entity Schema

This section describes how data entered in the OVBPI Modeler is defined in the database. For each business object there is a corresponding table in the dynamic schema. This table is created according to the guidelines described in the following sections.

## Data item names

The names for columns in the schema are based on the values you enter as the data items names in the Modeler. When added to the database, any illegal characters are replaced using a unicode representation, for example, `Orderu20Amount`, is a representation of Order Amount, where the `u20` is the hexadecimal for space.

If you enter u*nn* (where *nn* is a number) as part of a data item, be aware that this could cause data clashes with another data item that includes the same string as unicode character replacement.

## Data types

The dynamic schema uses a subset of data types used in the Flows schema; see Table Oracle and SQL Data Types on page 107 for the complete list.

## Keys

The tables that are generated have internally-generated Primary keys defined to assist in referential integrity management and for performance reasons. Referential integrity is a feature that prevents users or applications from entering inconsistent data.

## Data Objects

The Data_Objects table lists the types of Business Data definition that have been defined using the Modeler, plus statistics and statuses, which are calculated for these objects. The Primary Key is the Model_ID. Each Data Definition has a corresponding Instance table, which lists the instances of that object.

**Table 29 Data_Objects**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| Model_ID | string | 36 | No | Primary Key. |
| Name | string | 120 | No | The Data definition name. |
| Description | string | 256 | Yes | The description of the Data definition. |
| AvrgTime | float | | Yes | Average lifetime of instances of this data type in milliseconds. |
| SampleCount | integer | | Yes | Internal value used for calulating the average time, |
| ActiveInstances | integer | | Yes | Current number of instances. |
| TotalInstances | integer | | Yes | Total number of instances. |
| Status | string | 12 | Yes | The current status of the data object, which can be `Active` or `Deleted`. |
| RepositoryId | string | 36 | Yes | A unique identifier that can be used to associate different versions of the same definition; different versions of a Flow definition or Data definition have different Flow_IDs or Model_IDs but they have the same RepositoryId. |
| InstanceTable | string | 30 | Yes | The name of the database table used to store instances of this Data definition. |

**Table 29 Data_Objects**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| Subclass | string | 256 | Yes | For internal use. This column contains the name of the Data definition instance table. |
| Repository Revision | integer | | Yes | Revision of Data definition in the Model Repository. This revision information is displayed using the Repository Explorer. |
| FolderPath | string | 604 | No | For internal use. |

## Data Definition Instances

The Data definition instance table cannot be named or fully specified as it depends on the data entered using the Modeler; however, some columns appear in every Data definition instance and these are listed in the following table.

The name of this table is referenced in the `InstanceTable` column of the `Data_Objects` table when the Data definition is deployed within the Modeler.

**Table 30 Data Definition Instance**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| StartTime | date | | Yes | The time that this instance was started. |
| StartTimeLong Millis | long | | Yes | The time (in milliseconds) that this instance was started. |
| EndTime | date | | Yes | The time (in milliseconds) that this instance completed. |
| EndTimeLong Millis | long | | Yes | The time (in milliseconds) that this instance completed. |

**Table 30 Data Definition Instance**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| Status | string | 12 | Yes | The current status of the instance, which can be one of `Active` or `Completed`. |
| Id | string | 36 | No | Primary Key[a]. |
| Model_ID | string | 36 | Yes | Foreign Key[b]to Data_Objects table (Model_ID). |

a. This is the column that uniquely identifies rows.

b. This is the column that exists as the primary key in another table

# Business Event Handler Schemas

The following schemas are defined for the Business Event Handler. These schemas are used for the Event Store and for the openadaptor patients that are stored in the event hospital; the Event Hospital.

The tables structures are defined by openadaptor and in this context, a patient is an OVBPI event, which is in an openadaptor format as it is passed through openadaptor.

## Business Event Handler Event Store

The Event Store is an openadaptor Sink component that persists messages to a database repository. The messages are stored in the table in a string or equivalent format using Data Object XML format.

The Event Store is a persistent store that can be used to either:

- Take a copy of every event for archiving or auditing purposes
- Provide persistence in the Event Layer

Note that the Event Store can be use in combination with Hospitals; see the *OpenView Business Process Insight Integration Training Guide - Business Events* for details of the Event Store and how it can be used in conjunction with the Event Hospital.

**Table 31 EVENT_STORE**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| GUID | string | 256 | No | A unique identifier. |
| EVENT_GROUP | string | 256 | Yes | The event group that this event belongs to. Event grouping is a way of logically organizing events. An event group might be a path containing slashes (/) for further subgrouping. |
| EVENT_NAME | string | 256 | Yes | The event name. |

**Table 31 EVENT_STORE**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| DOXML | Variable length text | up to 2GB | No | The message. |
| STATUS | string | 100 | Yes | The status of the event, which can be one of `Unsent` or `Sent`. |
| TIME_RX | date | | No | Time when the event details are written to the Event Hospital. |
| TIME_TX | date | | Yes | Time when the event details are retrieved from the Event Hospital. |

## Event Hospitals

Event hospitals, can also be used to provide persistence. Event Hospitals are used to store events when an event can not be delivered to OVBPI for some reason. This might be because the event is invalid, or because the Business Impact Engine is offline.

**Table 32 DBusMH_Patient**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| PatientId | integer | | No | A unique identifier for each patient added to the event hospital. The identifier starts at 1 for the first event. |
| PatientStatus | string | 20 | No | The status of this patient, which can be one of:<br>• `New`, a newly added patient<br>• `Treated`, this patient message has been updated<br>• `Examined`, this patient message has been examined<br>• `Discharge_Wait`, this patient is waiting to be discharged<br>• `Dishcharge`, this patient has been discharged<br>• `Re_News`, this patient has been re-added to the hospital<br>• `Dead`, this patient has an unrecoverable error |
| DestAppName | string | 60 | No | The destination of this patient message; within OVBPI, this value is set to `BIA_app`. |
| Subject | string | 255 | Yes | The subject of this patient message; within OVBPI, this value is set to `BIA_subject`. |

**Table 32 DBusMH_Patient**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| RejectReason | string | 255 | Yes | The PipelineException message content, that is, the result of PipelineException.getMessage() from Java. |
| SourceAppName | string | 60 | Yes | The publisher of this patient message. |
| PubUniqueId | string | 255 | Yes | This is reserved for the publisher's unique message identifier. |
| IndexParam1 | string | 60 | Yes | Contains attributes of this patient data object and is used for performance efficiencies when unpacking the patient data object. |
| IndexParam2 | string | 60 | Yes | Contains attributes of this patient data object and is used for performance efficiencies when unpacking the patient data object. |
| IndexParam3 | string | 60 | Yes | Contains attributes of this patient data object and is used for performance efficiencies when unpacking the patient data object. |
| IndexParam4 | string | 60 | Yes | Contains attributes of this patient data object and is used for performance efficiencies when unpacking the patient data object. |
| Admitted | date | | No | The date that this patient was given the status `New`. |
| Examined | date | | Yes | The date that this patient was given the status `Examined`. |
| Treated | date | | Yes | The date that this patient was given the status `Treated`. |

**Table 32 DBusMH_Patient**

| Column Name | Data Type | Length | Allow Nulls | Description |
|---|---|---|---|---|
| Discharged | date | 8 | Yes | The date that this patient was given the status `Discharged`. |
| MarkedDead | date | 8 | Yes | The date that this patient was given a status of `Dead`. |
| Retried | integer | | Yes | The number of times the hospital has tried to send a message that has been marked for `DISCHARGE` back into the OVBPI system. |
| AdmittedUser | integer | | No | The database user that admitted this patient. |
| ExaminedUser | integer | | Yes | The database user that marked this patient as `Examined`. |
| TreatedUser | integer | | Yes | The database user that marked this patient as `Treated`. |
| DischargedUser | integer | | Yes | The database user that marked this patient as `Discharge_Wait` or `Discharge`. |
| MarkedDeadUser | integer | | Yes | The database user that marked this patient as `Dead`. |
| Patient | Variable length text | up to 2GB | Yes | The openadaptor data object that has been added to the hospital in XML format. |

# Complete List of OVBPI Database Tables

The following tables list the all the database tables and views that are created for OVBPI. Many of these tables are for internal use only and should not be modified.

This is list is provided in order that you can make sure that all the tables are included in any backup and recovery procedures and for you to check for any potential database table name clashes.

The capitalization used in the table is for ease of reference only. How the table names are represented in your database is database specific; for example, for an Oracle Server all table and view names are upper case.

Table 33 on page 166 lists all the OVBPI database tables and Table 34 on page 169 lists all the views.

**Table 33 Complete List of OVBPI Database Tables**

| Table Name | Internal Use Only (yes/no) |
|---|---|
| Arcs | no |
| BCE_Metric_Id_Gen | yes |
| DataInstIdToBeDeleted | yes |
| Data_Objects | no |
| DBusmh_Attribute | yes |
| DBusmh_EditableAttribute | yes |
| DBusmh_Patient | no |
| DBusmh_Role | yes |
| DBusmh_User | yes |
| DBusmh_UserRole | yes |
| Event_Store | no |
| FlowDataFilters | yes |
| FlowInstIdToBeDeleted | yes |

**Table 33 Complete List of OVBPI Database Tables**

| Table Name | Internal Use Only (yes/no) |
|---|---|
| FlowNodeZones | yes |
| Flows | no |
| Flow_Instance | no |
| ImpactThreshold | yes |
| Metric_CustomTypes | no |
| Metric_Definitions | yes |
| Metric_Dim_Dates | no |
| Metric_Dim_Flows | no |
| Metric_Dim_Flow_Instances | no |
| Metric_Dim_Groups | no |
| Metric_Dim_Metrics | no |
| Metric_Dim_Thresholds | no |
| Metric_Events | yes |
| Metric_Fact_Alerts | no |
| Metric_Fact_Statistics | no |
| Metric_Fact_Values | no |
| Metric_Generate_Errors | yes |
| Metric_Id_Gen | yes |
| Metric_Notification_Check_Time | yes |
| Metric_Staging_Statistics | yes |
| Metric_Stored_Procedures[a] | yes |
| NodeIdAndResourceStatus | yes |
| NodeInstance_Id_Gen | yes |

**Table 33 Complete List of OVBPI Database Tables**

| Table Name | Internal Use Only (yes/no) |
|---|---|
| NodeInstToBeDeleted | yes |
| Nodes | no |
| Nodes2Resources | no |
| Node_Instance | no |
| Node_Instance_CompletedTimes | no |
| Node_Instance_StartedTimes | no |
| NS_Digesterstore | yes |
| NS_DigesterSubscriptions | yes |
| NS_EmailRetry | yes |
| NS_EmailSubscriptions | yes |
| NS_EmailUsers | yes |
| NS_OVORetry | yes |
| NS_OVOSubscriptions | yes |
| NS_ScriptRetry | yes |
| NS_ScriptSubscriptions | yes |
| NS_SLAEmailSubscriptions | yes |
| NS_SLOEmailSubscriptions | yes |
| OVIS_AlarmStatus | yes |
| OVISTimeStamps | yes |
| Repos_defns | yes |
| Repos_Folders | yes |
| Repos_Labels | yes |
| Repos_Labels2Defns | yes |

**Table 33 Complete List of OVBPI Database Tables**

| Table Name | Internal Use Only (yes/no) |
|---|---|
| Resources | no |
| RMIEventRetry | yes |
| SOAPEventRetry | yes |
| Version | yes |

a. this table is available only with Microsoft SQL Server.

The following table lists all the OVBPI database views.

**Table 34 Complete List of OVBPI Database Views**

| View Name | Internal Use Only (yes/no) |
|---|---|
| Dbusmh_View | yes |
| Metrics | no |
| Metric_Values | no |
| Repos_Definitions | yes |

The following table lists the OVBPI database indexes.

**Table 35 Complete List of OVBPI Database Indexes**

| Index Name | Internal Use Only (yes/no) |
|---|---|
| Flow_Instance_Idx_*nnn* | yes |
| Idx*n* | yes |
| Metric_Dim_Dates_Idx_*nnn* | yes |
| Metric_Fact_Groups_Idx_*nnn* | yes |
| Metric_Fact_Alerts_Idx_*nnn* | yes |
| Metric_Fact_Stats_Idx_*nnn* | yes |
| Metric_Fact_Values_Idx_*nnn* | yes |

**Table 35 Complete List of OVBPI Database Indexes**

| Index Name | Internal Use Only (yes/no) |
|---|---|
| Metric_Staging_Statistics_Idx_*nnn* | yes |
| Node_Instance_Comple_Idx_*nnn* | yes |
| Node_Instance_Idx_*nnn* | yes |
| Node_Instance_Starte_Idx_*nnn* | yes |
| Sys_*string* | yes |

There are also stored procedures, database triggers and other database scripts defined by OVBPI and potentially defined by you. These also need to be taken into account for backup purposes or if you need to move the OVBPI data.

It is strongly recommended that you create an Oracle User or Microsoft SQL Server Database specifically for OVBPI as it makes the task of identifying this OVBPI data much easier.

# B Expression Grammar in Flow, Data and Filter Definitions

This appendix lists the rules for the grammar that can be used for business flow progression rules and expressions, and expressions within business process metric filter definitions.

The progression rules and expressions are within the OVBPI Modeler for the start and complete conditions for nodes, and for filter expressions for Data definitions when subscribing to events.

Business process metric filter expressions are used within the Metric definer to optionally specify the conditions for collecting statistical data for the business process metrics.

The grammar used for expressions is similar to Java expressions. The following sections describe the grammar and its construct. Further examples of using this grammar are provided in the *OpenView Business Process Insight Integration Training Guide - Modeling Flows*.

# Grammar

This following is an informal description of the grammar, it shows how expressions can be constructed. Note that this construct is subject to specific limitations according to how it is being used. Examples of the use of the grammar are provided in the following sections.

```
expression =>
 property-value = expression
|expression + expression
|expression - expression
|expression * expression
|expression / expression
|expression % expression
|- expression
|expression && expression
|expression || expression
|! expression
|expression == expression
|expression > expression
|expression >= expression
|expression < expression
|expression <= expression
|expression != expression
|(expression)
|expression ? expression : expression
| value

value =>
       constant_value
      |property_value
      |function_return_value
```

See section Function Return Values on page 174 for possible values for `function_return_value`.

```
constant_value =>
       string_contant
      |integer_constant
      |real_constant
      |true
      |false
      |null

property_value =>
       root_object.relationship_name.property_name
```

where:

- `root_object` is `this` or `event`.

  If no `root_object` is specified, the expression assumes `this`.

- `relationship_name` can be included zero or more times, depending on how the property relates to the root object.

Possible `property_value` expressions are:

- `this.property`
- `this.data.property`
- `event.property`
- `property`

The following are examples that you might use in your expressions. Further examples are given in the *OpenView Business Process Insight Integration Training Guide - Modeling Flows*:

- `this.OrderNumber`
- `this.Customer.Customer_ID`

# Function Return Values

The following sections describe the data and string values that can be returned from the `value` expression.

These functions can be used in filter expressions, binding expressions, progression rules and assignment actions.

## Functions for Dates and Times

The following functions convert expressions into milliseconds:

- `hours (`*`number`*`)`
- `minutes (`*`number`*`)`
- `seconds (`*`number`*`)`
- `days (`*`number`*`)`

These functions convert *`number`* to the equivalent number of milliseconds (which are the Business Impact Engine time units).

## Functions for Identifying String Values

The following functions take a string parameter and return a boolean result.

- `string_property.contains(`*`value`*`)`

  Returns `true` if *`value`* is found in the property value .

- `string_property.starts(`*`value`*`)`

  Returns `true` if the property value starts with *`value`*.

- `string_property.ends(`*`value`*`)`

  Returns `true` if the property value ends with *`value`*.

## Functions for All Property Types

The following function takes one or more parameters and returns a boolean result:

`property.in(`*`value`*`,`*`value`*`,...)`

This expression returns `True` if the value of `property` is one of the listed set of values (`(`*`value, value,...`*`)`, and `False` if it is not.

# Flow Progression Rules

The grammar for flow progression rules is slightly different from the simple expression grammar, as it is used to describe deltas or changes to Data definition properties.

There are four styles of progression rule provided in OVBPI as follows:

- Complete on first assignment
- Complete on transition
- Start and complete on transitions
- Advanced conditions

The Advanced conditions style of progression rule requires you to enter the methods for the progression rules directly as described in section Methods for Progression Rules on page 178.

For the remaining progression styles, you do not need to enter the methods, you just select the style of the progression rule that you want for the node. Ultimately, each style uses a subset, or selection of the methods described in section Methods for Progression Rules on page 178. You can see these methods if you define a progression rule using one of the styles and then change the style to Advanced conditions to view the methods created for the style.

More detailed information about using these progression rules in your flows is provided in the *OpenView Business Process Insight Integration Training Guide - Modeling Flows*.

The methods used for the progression rule styles are listed in Table 36 on page 176.

**Table 36 Methods Used for Progression Rule Styles**

| Style | Methods Used | Comments |
|---|---|---|
| Complete on first assignment | `before()==null` | This style of progression rule uses the method listed to determine when the property value changes from Null to any other value. |

**Table 36 Methods Used for Progression Rule Styles**

| Style | Methods Used | Comments |
|---|---|---|
| Complete on transition | Complete Condition: `before().in()` and `after().in()` | This style of progression rule uses the methods listed to determine when the node complete condition is met. It does this by testing the original and modified value of a property against one or more values in the rule. Use of the `in()` method enables you to enter a selection of values that can be tested. If you do not enter a value, any value for the property can satisfy the condition. |
| Start and complete on transitions | Start Condition: `before().in()` and `after().in()` Complete Condition: `before().in()` and `after().in()` | This style of progression rule uses the methods listed to determine when the node start and node complete conditions are met. It does this by testing the original and modified value of a property against one or more values in the rule. Use of the `in()` method enables you to enter a selection of values that can be tested. If you do not enter a value, any value for the property can satisfy the condition. |

## Methods for Progression Rules

In order to express progression rules using the `Advanced Conditions` option in the OVBPI Modeler, there are methods for Data definitions and their properties. A method represents the value relating to the data object or property at a point in time.

The methods are described in the following table and apply only to a property in a flow progression rules.

**Table 37 Methods for Progression Rules**

| Method | Description |
|---|---|
| `this.data.property.changed()` | This function is used to test when the property value changes. When the property value changes it returns a result of `True`. |
| `this.data.property.before()== "prop-value"` | This evaluates to the value of the property before a change and is executed only when the property values changes. It is implicit in this expression that the property value was as stated by *`prop-value`* and is now no longer that value.<br>The type returned is the same as the type of the property. |
| `this.data.property.after()== "prop-value"` | This evaluates to the value of the property after a change and is executed only when the property value changes. It is implicit in this expression that the property value was a value other than *`prop-value`* and is now *`prop-value`*. The type returned is the same as the type of the property. |

**Table 37 Methods for Progression Rules**

| Method | Description |
|---|---|
| `this.data.created()` | This evaluates to true when the data definition is created. This method is executed only when a new Data definition is created.<br>The method returns a Boolean result. |
| `this.data.property.in( "prop-value1", "prop-value2", ...)` | This evaluates to true when the value of the propery is set to any of the listed values.<br>The method returns a Boolean result. |
| `this.data.terminated()` | This evaluates to true when the data definition is terminated, specifically when the Data definition received an event that is flagged as terminating it. This is specified in the Event Subscription dialog for the Data Definition; `Terminate this instance of the Data Definition after handling the event`.<br>The method returns a Boolean result. |

## Example of Complete Condition for Start Node When Data Definition Created

The following shows an example of the expression that you use when you want a flow instance to be started when a Data definition for the flow is created. In this case, a new flow instance is started only when the start and complete conditions for the start node in the flow are met, for example:

```
this.order.created()
```

where:

- `this` is the reference to the flow, in this case the `Order` flow.
- `order` is the Data definition.
- `created()` is a method that evaluates to true when the `order` Data definition is created.

## Example Start Condition for Start Node When Data Definition Modified

The following shows an example of the expression that you use when you want a flow instance to be started when the Data definition for the flow is modified. In this case a flow instance is started not when the Data definition is created but when there is a change to the Data definition property.

```
this.order.priority.before() == null &&
this.order.priority.after() <= 3
```

where:

- `this` is the reference to the flow, in this case the `Order` flow.
- `order` is the primary Data definition.
- `priority` is an integer property.
- `before()` is a method, which returns the value of the property before the change.
- `null` indicates that the property was not initialized before the change.
- `after()` is a method, which returns the value of the property after the change.

## Example Start Condition using a Method on a Method

The following example is similar to the example above; however, shows an example of the expression that you use when you want a flow instance to be started when the Data definition for the flow is modified and the Data definitions contains a specific string.

```
this.order.customer_id.after().contains("X")
```

where:

- `this` is the reference to the flow, in this case the `Order` flow.
- `order` is the primary Data definition.
- `customer_id` is a string property that is a unique identifier for the customer.
- `after()` is a method, which returns the value of the property after the change.
- `contains()` is a string function, which returns a boolean result depending on whether the string "`X`" is present in the `customer_id`.

# Case Sensitivity for Expressions

This section describes case sensitivity for expression properties and for expressions containing string constants.

## Expression Properties

Properties and methods in expressions are case sensitive, which means the following expression are not equivalent within your OVBPI system:

- `this.order.priority.before() == null && this.order.priority.after() <= 3`
- `this.Order.Priority.before() == null && this.Order.Priority.after() <= 3`

## Expressions with String Constants

Expressions containing string constants are case sensitive, which means the following expressions are not equivalent within your OVBPI system:

- `this.order.status.after() == "a"`
- `this.order.status.after() == "A"`

# C Coercion Rules

This appendix describes the rules for how properties are coerced when evaluating binding, filter and assignment expressions. This is an enforced evaluation by the OVBPI Modeler, based on the rules described in this Appendix.

Valid data types for use within OVBPI are:

- String
- Numeric, which can be one of:
  - Integer
  - Long
  - Double
  - Currency
- Boolean
- Date

# Assignments

The following table shows how assignments are coerced:

**Table 38 Assignment Coercion**

| Data Type... | Assigned from... |
|---|---|
| `String` | Any type |
| `Numeric` | Any other numeric value (with possible truncation if assignment is to an `integer`), or from `Strings` provided that the string is numeric. If the assignment does not produce a valid number, you receive an error similar to the following in the Business Impact Engine log file:<br>`Cannot assign value` *`prop-name`* `to property` *`prop-name`* `as it is not a compatible type.` |
| `Boolean` | Other Boolean values, or from `Strings` provided that the string is either “true” or “false”. If the assignment does not produce a valid boolean expression, you receive an error similar to the following in the Business Impact Engine log file:<br>`Cannot assign value` *`prop-name`* `to property` *`prop-name`* `as it is not a compatible type.` |
| `Date` | Other `Dates` or a `Numeric` value (treated as number of milliseconds since 1970). |

# Expressions

The following table shows how values within expressions are coerced.

**Table 39**

| Operator | Behavior |
|---|---|
| Arithmetic:<br>• +<br>• –<br>• *<br>• /<br>• % | Supported between Numeric properties or values that can be coerced to numeric using the assignment rules described in Table 38. |
| Comparison:<br>• <<br>• <=<br>• =<br>• !=<br>• ><br>• >= | The comparison operators can be used to provide a:<br>• numeric comparison between Numeric properties or values that can be coerced to numeric using the assignment rules described in Table 38.<br>• date comparison between Date properties.<br>• alphabetic comparison (locale-specific) between String properties or values that can be coerced to Strings using the using the assignment rules described in Table 38 (including Boolean).<br>The result of a comparison is always a boolean value. |
| String tests:<br>• starts()<br>• ends()<br>• contains()<br>• in() | Performs a test on a String property.<br>Note that these tests cannot be applied to non-string properties. |

**Table 39**

| Operator | Behavior |
|---|---|
| Logical:<br>• `!` (Not)<br>• `&&` (And)<br>• `\|\|` (Or) | Supported between Boolean properties or values that can be coerced to boolean using the assignment rules described in Table 38. |
| Conditional value `if, then, else:`<br>• `?:` | First operand must be Boolean or be coerced to boolean, other operands can be any type; however they must be the same each other. |

## Using NULL Within an Expression

The OVBPI Modeler allows only the `equals` and `not-equals` tests with the `null` constant. However, there might be expressions using other operators involving values that are `null` when these expressions are executed, for example how is the following expression evaluated when *index* is `null`?

```
index + 1
```

The following table shows the rules that apply when comparing data items within an expression where one data item contains a `null` value.

**Table 40**

| Operator | Comparison |
|---|---|
| • ==<br>• != | Returns TRUE if the other operand is (or is not) `null`. |
| All other comparison operators | Always return `FALSE`.<br>For example the following expressions both return `FALSE` if `index` is `null`:<br>• `index > 10`<br>• `index <= 10` |
| Arithmetic operators | Arithmetic operators for expressions containing a null value always return null as the result, for example:<br>`index + 1` returns `null` if `index` is equal to `null` |

Note that the `null` constant is case sensitive and when used in Java expressions, should always be lower case.

# Index

## A


ACTIVE_INSTANCES_HIGH, 80, 83

ACTIVE_INSTANCES_LOW, 80, 83

ACTIVE_VALUES_HIGH, 81, 84

ACTIVE_VALUES_LOW, 80, 83

ActiveCount
  Nodes table, 146

ActiveFlows
  Flows table, 140

ActiveInstances
  Data_Objects table, 158

ActiveWeight
  Nodes table, 147

Adapter
  Business Event Handler, 37
  for iWay integration, 56
  for OVBPI Server, 26

Administration Console
  architecture, 42
  OVBPI Notification Server, 42
  OVBPI Server Administration Console, 42

Admitted
  DBusMH_Patient table, 164

AdmittedUser
  DBusMH_Patient table, 165

Advanced conditions
  flow progression rules, 176
  grammar rules, 176

Alarms and SLOs
  configuring, 79

AlertCriticalLevel
  Metric_Dim_Thresholds table, 132

AlertCriticalLevelDefnUnit
  Metric_Dim_Thresholds table, 133

Alert facts dimension, 108

AlertLevel
  Metric_Fact_Alerts table, 111

AlertMajorLevel
  Metric_Dim_Thresholds table, 132

AlertMajorLevelDefnUnit
  Metric_Dim_Thresholds table, 133

AlertMinorLevel
  Metric_Dim_Thresholds table, 132

AlertMinorLevelDefnUnit
  Metric_Dim_Thresholds table, 133

AlertStatus
  Metric_Fact_Alerts table, 111

AlertWarningLevel
  Metric_Dim_Thresholds table, 132

AlertWarningLevelDefnUnit
  Metric_Dim_Thresholds table, 133

Architecture
Administration Console, 42
and OVBPI database, 18
Business Event Handler, 36
Business Impact Engine, 24
Business Process Dashboard, 13
diagram for OVBPI, 12, 46
Intervention Client, 16
Metric definer, 28
Metric Engine, 30
Notification Server, 33
OVBPI, 11, 45
OVBPI OpenView Operations Adapter, 52
OVBPI Probes and alarms, 50
OVBPI Server, 22
Repository Explorer, 32
Security, 43
Self-Healing Services, 59
Arcs table, 152
Destination, 152
Flow_ID, 152
Source, 152
Type, 152
Arithmetic operator
coercion, 185
Assignment
coercion and data types, 184
expressions and coercion rules, 183
AVAILABILITY, 80, 82, 83
AverageValue
Metric_Fact_Statistics table, 119
AvrgTime
Data_Objects table, 158
Flows table, 140
Nodes table, 147

## B

BIAEngineAdaptor
Engine Adapter, 38
Binding expressions
coercion rules, 183
Boolean
data type coercion rule, 184
BPEL
importing into Model Repository, 40
Building applications
using OVBPI data, 106
Business Entity schema, 106, 157
Data_Object-Instance table, 159
Data_Objects table, 158
data item names, 157
Business Event Handler
architecture, 36
event adapter, 37
openadaptor, 36
OVBPI Server component, 22
receiving events, 38
schema, 106, 161
Business Impact Engine
architecture, 24
BIAEngineAdaptor, 38
Business Metrics
architecture diagram for Business Process Metric definer, 29
schema, 105, 108
Business object manager
OVBPI Server, 26
Business Process Dashboard
and business process metrics, 15
architecture, 13
JSPs, 15

Business process metrics
Business Process Metric definer an OVBPI Server component, 28
data in the OVBPI database, 20

## C

C_OVBPI_FLOW_PROBE
ACTIVE_INSTANCES_HIGH, 80
ACTIVE_INSTANCES_LOW, 80
ACTIVE_VALUES_HIGH, 81
ACTIVE_VALUES_LOW, 80
AVAILABILITY, 80
INSTANCE_TPUT_HIGH, 81
INSTANCE_TPUT_LOW, 81
objective information, 80
RESPONSE_TIME, 80
SETUP_TIME, 80
TRANSFER_TPUT, 80
VALUE_TPUT_HIGH, 81
VALUE_TPUT_LOW, 81

C_OVBPI_FLOW OVBPI_DB_Password
OVBPI_DB_Password, 74

C_OVBPI_FLOW - OVBPI_FLOW probe, 72

C_OVBPI_FLOW probe
OVBPI_DB_User_Name, 73
OVBPI_Flow_Name, 74
OVBPI_Identifier, 73
Port, 73
Target Host, 73
Username, 73

C_OVBPI_METRIC_PROBE
AVAILABILITY, 82
objective information, 82
RESPONSE_TIME, 82
SETUP_TIME, 82
TBN_DURATION_HIGH, 82
TBN_DURATION_LOW, 82
TRANSFER_TPUT, 82

C_OVBPI_METRIC - OVBPI_METRIC probe, 72

C_OVBPI_METRIC probe
OVBPI_DB_Password, 76
OVBPI_DB_User_Name, 75
OVBPI_Flow_Name, 76
OVBPI_Identifier, 75
OVBPI_Metric_Name, 76
Port, 75
Target Host, 75
Username, 75

C_OVBPI_NODE_PROBE
ACTIVE_INSTANCES_HIGH, 83
ACTIVE_INSTANCES_LOW, 83
ACTIVE_VALUES_HIGH, 84
ACTIVE_VALUES_LOW, 83
AVAILABILITY, 83
INSTANCE_TPUT_HIGH, 84
INSTANCE_TPUT_LOW, 84
objective information, 83
RESPONSE_TIME, 83
SETUP_TIME, 83
TRANSFER_TPUT, 83
Username, 77
VALUE_TPUT_LOW, 84

C_OVBPI_NODE - OVBPI_NODE probe, 72

C_OVBPI_NODE probe
OVBPI_DB_Password, 77
OVBPI_DB_User_Name, 77
OVBPI_Flow_Name, 78
OVBPI_Identifier, 77
OVBPI_Node_Name, 78
Port, 77
Target Host, 77

Case sensitivity
for expressions within progression rules, 182

Coercion
arithmetic expressions, 185
comparison expressions, 185
conditional expressions, 186
for expressions, 185
logical expressions, 186
rules for binding, filter and assignment expressions, 183
string expressions, 185

Comparison operator
coercion, 185

Complete condition
after(), 178
before(), 178
changed(), 178
created(), 179
in(), 179
methods for, 178
terminated(), 179

CompletedTime
Node_Instance_CompletedTimes table, 151

CompletedTimeLongMillis
Node_Instance_CompletedTimes table, 151

Complete on first assignment
grammar rules, 176

Complete on transition
grammar rules, 176

Component
Deployer, 41
Model Repository, 40

Conditional operator
coercion, 186

Configuring
OpenView Dashboard integration, 57
OVIS alarms and SLOs, 79

Console
Notification Server Administration Console, 41
OVBPI Administration Console, 41

CountTotal
Metric_Fact_Statistics table, 118

CreatedDate
Metric_Dim_Metrics table, 121
Metric_Dim_Thresholds table, 133

Currency
data type definition, 107

CurrentAlertFromTime
Metric_Dim_Thresholds table, 132

CurrentAlertLevel
Metric_Dim_Thresholds table, 132

CurrentAlertStatus
Metric_Dim_Thresholds table, 132

Custom
fields and defining for OVSD, 92
probes and using with OVIS, 69

CustomMetricDescription
Metric_CustomTypes table, 134

CustomMetricName
Metric_CustomTypes table, 134

CustomSPName
Metric_CustomTypes table, 134

## D

Dashboard
and business process metrics, 15
and OpenView Dashboard, 57
architecture, 13
JSPs, 15

Data_Object-Instance table, 159
EndTime, 159
EndTimeLongMillis, 159
Id, 160
Model_ID, 160
StartTime, 159
StartTimeLongMillis, 159
Status, 160
Data_Objects table
ActiveInstances, 158
AvrgTime, 158
Business Entity schema, 158
Description, 158
FolderPath, 159
InstanceTable, 158
Model_ID, 158
Name, 158
RepositoryID, 158
RepositoryRevision, 159
SampleCount, 158
Status, 158
Subclass, 159
TotalInstances, 158
Database
schema, 105
DataDefinition_ID
Metric_Dim_Flow_Instances table, 126
Metric_Dim_Flows table, 125
DataDefinitionInstance_ID
Metric_Dim_Flow_Instances table, 126
Data item names
Business Entity schema, 157
Data type
assignment coercion, 184
Business Entity Schema, 157
Data type definition
Currency, 107
Date, 107
Float, 107
Integer, 107
Long, 107
OVBPI Schema, 107
String, 107
Text, 107
variable-length binary, 107
variable-length text, 107
Date
data type coercion rule, 184
data type definition, 107
functions, 174
Date_ID
Metric_Dim_Dates table, 122
Metric_Fact_Alerts table, 110
Metric_Fact_Statistics table, 117
Metric_Fact_Value table, 113
Day
Metric_Dim_Dates table, 123
DayOfWeek
Metric_Dim_Dates table, 123
DayOfWeek_Num
Metric_Dim_Dates table, 123
DBusMH_Patient
database table, 163

DBusMH_Patient table
Admitted, 164
AdmittedUser, 165
DestAppName, 163
Discharged, 165
DischargedUser, 165
Examined, 164
ExaminedUser, 165
IndexParam1, 164
IndexParam2, 164
IndexParam3, 164
IndexParam4, 164
MarkedDead, 165
MarkedDeadUser, 165
Patient, 165
PatientId, 163
PatientStatus, 163
PubUniqueId, 164
RejectReason, 164
Retried, 165
SourceAppName, 164
Subject, 163
Treated, 164
TreatedUser, 165
Deadline
Metric_Fact_Value table, 114
DeadlineOverdue
Metric_Fact_Value table, 115
Defining
expressions for flows, 171
DeploymentStatus
Resources table, 154
DestAppName
DBusMH_Patient table, 163
Destination
Arcs table, 152
Developing applications
using OVBPI data, 106
Dimension
alert facts, 108
schema, 108
statistics facts, 116
tables in database, 121
value facts, 112
Discharged
DBusMH_Patient table, 165
DischargedUser
DBusMH_Patient table, 165
DOXML
EVENT_STORE table, 162

## E

Email
notification data in OVBPI database, 21
Enable OVIS integration, 86
EndCondition
Metrics table, 136
EndNode
Metrics table, 136
EndTime
Data_Object-Instance table, 159
Flow_Instances table, 144
Metric_Dim_Flow_Instances table, 127
Metric_Values table, 137
Node_Instance table, 149
EndTimeLongMillis
Data_Object-Instance table, 159
Flow_Instances table, 145
Metric_Values table, 137
Node_Instance table, 149
Engine
OVBPI Server component, 22
Entity model schema, 108
Entity relationship diagram
OVBPI schema, 156

Event
hospital and database table, 163
hospital data and OVBPI database, 21
propagation, 38
receivers and transmitters for the OVBPI Server, 27
EVENT_GROUP
EVENT_STORE table, 161
EVENT_NAME
EVENT_STORE table, 161
EVENT_STORE
table, 161
EVENT_STORE table
DOXML, 162
EVENT_GROUP, 161
EVENT_NAME, 161
GUID, 161
STATUS, 162
TIME_RX, 162
TIME_TX, 162
Event adapter
Business Event Handler, 37
Event Handler
See Business Event Handler
Event Store
definition, 161
schema, 161
Examined
DBusMH_Patient table, 164
ExaminedUser
DBusMH_Patient table, 165
Example
start condition for Start Node
Data definition created, 180
Data definition modified, 180
start condition using a method on a method, 181
Expression
coercion, 185
grammar, 171
properties
case sensitivity for Flow definitions, 182
using NULLs, 187

## F

Filter expression
coercion rules, 183
Float
data type definition, 107
Flow
data and the OVBPI database, 20
expressions
functional return values, 174
progression rules
flow definitions, 176
schema, 105, 139
Flow_ID
Arcs table, 152
Flow_Instances table, 142
Flows table, 140
Metric_Dim_Flow_Instances table, 126
Metric_Dim_Flows table, 124
Metric_Dim_Metrics table, 121
Metric_Fact_Alerts table, 110
Metric_Fact_Statistics table, 117
Metric_Fact_Value table, 113
Metrics table, 135
Nodes table, 146
Flow_Instance
Metric_Values table, 137

Flow_Instances table, 145
EndTime, 144
EndTimeLongMillis, 145
Flow_ID, 142
FlowInstance_ID, 142
Identifier, 142
Primary_entity, 143
Primary_entity_ inst, 143
schema, 142
StartTime, 144
StartTimeLongMillis, 144
Status, 145
Weight, 143
Weight_type, 144
Flow definition, 171
expression grammar, 171
expression properties
case sensitivity, 182
functions for dates and times, 174
Modeler
date and time functions for Flow definitions, 174
Progression rules, 176
String values, 174
Time functions, 174
FlowDescription
Flows table, 140
FlowInstance_ID
Flow_Instances table, 142
Metric_Dim_Flow_Instances table, 126
Metric_Fact_Alerts table, 110
Metric_Fact_Value table, 113
Node_Instance table, 148
Flow Instances
C_OVBPI_FLOW - OVBPI_FLOW probe, 72
Flow Instances probe
OVBPI_DB_Password, 74
OVBPI_DB_User_Name, 73
OVBPI_Flow_Name, 74
OVBPI_Identifier, 73
Port, 73
Target Host, 73
Username, 73
Flow Metric probe
OVBPI_DB_Password, 76
OVBPI_DB_User_Name, 75
OVBPI_Flow_Name, 76
OVBPI_Identifier, 75
OVBPI_Metric_Name, 76
Port, 75
Target Host, 75
Username, 75
Flow Metrics
C_OVBPI_METRIC - OVBPI_METRIC probe, 74
FlowName
Flows table, 140
Metric_Dim_Flows table, 124
Flow probe
ACTIVE_INSTANCES_HIGH, 80
ACTIVE_INSTANCES_LOW, 80
ACTIVE_VALUES_HIGH, 81
ACTIVE_VALUES_LOW, 80
AVAILABILITY, 80
INSTANCE_TPUT_HIGH, 81
INSTANCE_TPUT_LOW, 81
RESPONSE_TIME, 80
SETUP_TIME, 80
TRANSFER_TPUT, 80
VALUE_TPUT_HIGH, 81
VALUE_TPUT_LOW, 81

Flows table, 140
ActiveFlows, 140
AvrgTime, 140
Flow_ID, 140
FlowDescription, 140
FlowName, 140
FolderPath, 142
Primary_entity, 141
RepositoryID, 141
RepositoryRevision, 142
SampleCount, 140
Status, 141
Subclass, 141
TotalFlows, 140
FolderPath
Data_Objects table, 159
Flows table, 142
Functional return values
flow expressions, 174

## G

Grammar rules, 171
Data definitions, 171
Flow definitions, 171
Metric definitions, 171
Group_ID
Metric_Dim_Groups table, 128
Metric_Fact_Statistics table, 118
Metric_Fact_Value table, 114
GroupName
Metric_Dim_Groups table, 128
Metric_Dim_Metrics table, 121
GroupValue
Metric_Dim_Groups table, 128
GUID
EVENT_STORE table, 161

## H

Hour
Metric_Dim_Dates table, 123

## I

Id
Data_Object-Instance table, 160
Identifier
Flow_Instances table, 142
Metric_Dim_Flow_Instances table, 126
Idx
Metric_Fact_Value table, 115
Metric_Value table, 138
Node_Instance_CompletedTimes table, 151
Node_Instance_StartedTimes table, 150
IndexParam1
DBusMH_Patient table, 164
IndexParam2
DBusMH_Patient table, 164
IndexParam3
DBusMH_Patient table, 164
IndexParam4
DBusMH_Patient table, 164
INSTANCE_TPUT_HIGH, 81, 84
INSTANCE_TPUT_LOW, 81, 84
InstanceRate
Nodes table, 147
InstanceTable
Data_Objects table, 158
Integer
date type definition, 107
Integrating with iWay, 56
Intervention Client
architecture, 16

IsAcknowledged
Metric_Fact_Alerts table, 111
IsDeleted
Metric_Dim_Metrics table, 122
IsLatest
Metric_Fact_Statistics table, 117
iWay integration, 56

## J

JSPs
used within Dashboard, 15

## K

Keys
Business Entity Schema, 157

## L

LastChange
Resources table, 154
LastChangeLongMillis
Resources table, 154
LastCheckTime
Metric_Dim_Thresholds table, 132
LastRateUpdate
Nodes table, 148
LastRateUpdateLongMillis
Nodes table, 148
LastStatisticsRecordTime
Metric_Dim_Metrics table, 122
LastUpdateTime
Metric_Fact_Value table, 115
Logical operator
coercion, 186
Long
data type definition, 107

## M

MarkedDead
DBusMH_Patient table, 165
MarkedDeadUser
DBusMH_Patient table, 165
MaximumValue
Metric_Fact_Statistics table, 119
MeasurementPeriod
Metric_Dim_Metrics table, 121
Metric_Fact_Statistics table, 118
Metric_Custom_Types table, 134
Metric_CustomTypes table
CustomMetricDescription, 134
CustomMetricName, 134
CustomSPName, 134
ValueUnits, 134
Metric_Dim_Dates table, 122
Date_ID, 122
Day, 123
DayOfWeek, 123
DayOfWeek_Num, 123
Hour, 123
Minute, 124
Month, 123
Quarter, 122
Time, 124
Week, 123
Year, 122
Metric_Dim_Flow_Instances table, 126
DataDefinition_ID, 126
DataDefinitionInstance_ID, 126
EndTime, 127
Flow_ID, 126
FlowInstance_ID, 126
Identifier, 126
StartTime, 127
Status, 127
WeightType, 126

Metric_Dim_Flows table, 124
Data_Definition_ID, 125
Flow_ID, 124
FlowName, 124
Status, 125
Metric_Dim_Groups table, 128
Group_ID, 128
GroupName, 128
GroupValue, 128
Metric_Dim_Metrics table, 121
CreatedDate, 121
Flow_ID, 121
GroupName, 121
IsDeleted, 122
LastStatisticsRecordTime, 122
MeasurementPeriod, 121
Metric_ID, 121
MetricName, 121
ValueUnits, 121
Metric_Dim_Thresholds table, 129
CreatedDate, 133
Metric_Dim_Threshold table
AlertCriticalLevel, 132
AlertCriticalLevelDefnUnit, 133
AlertMajorLevel, 132
AlertMajorLevelDefnUnit, 133
AlertMinorLevel, 132
AlertMinorLevelDefnUnit, 133
AlertWarningLevel, 132
AlertWarningLevelDefnUnit, 133
CurrentAlertFromTime, 132
CurrentAlertLevel, 132
CurrentAlertStatus, 132
LastCheckTime, 132
Metric_ID, 129
MetricThreshold_ID, 129
StagingAlertLevel, 133
ThresholdColumnName, 130
ThresholdDescription, 129
ThresholdMessage, 129
ThresholdName, 129
ThresholdTest, 131
ThresholdType, 130
Metric_Fact_Alerts table, 110
AlertLevel, 111
AlertStatus, 111
Date_ID, 110
Flow_ID, 110
FlowInstance_ID, 110
IsAcknowledged, 111
MetricAlert_ID, 110
MetricThreshold_ID, 110
RaisedTime, 110
Time, 110
Value, 111

Metric_Fact_Statistics table, 117
AverageValue, 119
CountTotal, 118
Date_ID, 117
Flow_ID, 117
Group_ID, 118
IsLatest, 117
MaximumValue, 119
MeasurementPeriod, 118
Metric_ID, 117
MetricStatistic_ID, 117
MinimumValue, 119
StatisticsType, 118
StdDevValue, 119
SumSquareValue, 119
SumValue, 118
Throughput, 118
Time, 117
WeightAverageValue, 120
WeightStdDevValue, 120
WeightSumSquareValue, 119
WeightSumValue, 119
WeightThroughput, 119
WeightTotal, 119
Metric_Fact_Values table, 113
MetricValue_ID, 113
Metric_Fact_Value table
Date_ID, 113
Deadline, 114
DeadlineOverdue, 115
Flow_ID, 113
FlowInstance_ID, 113
Group_ID, 114
Idx, 115
LastUpdateTime, 115
Metric_ID, 113
StartTime, 115
Status, 114
TimeCompleted, 114
Value, 115
Weight, 114
Metric_Generate_Errors table, 135
Metric_ID
Metric_Dim_Metrics table, 121
Metric_Dim_Thresholds table, 129
Metric_Fact_Statistics table, 117
Metric_Fact_Value table, 113
Metric_Values
Flow_Instance, 137
table
EndTime, 137
EndTimeLongMillis, 137
Idx, 138
Name, 137
StartTime, 137
StartTimeLongMillis, 137
Time, 138
Type, 137
Value, 137
Metric_Values table, 137
MetricAlert_ID
Metric_Fact_Alerts table, 110
Metric data
OVBPI database, 20
Metric definer
architecture, 28
Metric Engine
architecture, 30
OVBPI Server, 30
MetricName
Metric_Dim_Metrics table, 121
Metrics table, 135
Metric probe
AVAILABILITY, 82
RESPONSE_TIME, 82
SETUP_TIME, 82
TBN_DURATION_HIGH, 82
TBN_DURATION_LOW, 82
TRANSFER_TPUT, 82

Metrics table
EndCondition, 136
EndNode, 136
Flow_ID, 135
MetricName, 135
MetricType, 135
StartCondition, 136
StartNode, 136
MetricStatistics_ID
Metric_Fact_Statistics table, 117
MetricThreshold_ID
Metric_Dim_Thresholds table, 129
Metric_Fact_Alerts table, 110
Metric threshold data
OVBPI database, 21
Metric thresholds
grammar rules, 171
MetricType
Metrics table, 135
MetricValue_ID
Metric_Fact_Values table, 113
MinimumValue
Metric_Fact_Statistics table, 119
Minute
Metric_Dim_Dates table, 124
Model_ID
Data_Object-Instance table, 160
Data_Objects table, 158
Modeler
Advanced conditions grammar, 176
Complete on first assignment grammar, 176
Complete on transition grammar, 176
Flow definition language, 171
functional return values for flow definitions, 174
grammar description, 172
grammar rules, 171
progression rules
defining for flows, 176
Start and complete on transition grammar, 176
String values for Flow definitions, 174
Model Repository
data and the OVBPI database, 21
description, 40
Monitoring OVSD processes, 53
Month
Metric_Dim_Dates table, 123

## N

Name
Data_Objects table, 158
Metric_Values table, 137
Node2Resources
Resource_ID, 154
table
Node_ID, 154
Node2Resources table, 154
Node_ID
Node2Resources table, 154
Node_Instance table, 148
Nodes table, 146

Node_Instance_CompletedTimes table, 151
CompletedTime, 151
CompletedTimeLongMillis, 151
Idx, 151
NodeInstance_ID, 151
Node_Instance_StartedTimes
NodeInstance_ID, 150
table, 150
Idx, 150
StartedTime, 150
StartedTimeLongMillis, 150
Node_Instance table
EndTime, 149
EndTimeLongMillis, 149
FlowInstance_ID, 148
Node_ID, 148
NodeInstance_ID, 148
ResourceStatus, 149
StartTime, 149
StartTimeLongMillis, 149
Status, 149
Table
Node_Instance, 148
NodeDescription
Nodes table, 146
NodeInstance_ID
Node_Instance_CompletedTimes table, 151
Node_Instance_StartedTimes, 150
Node_Instance table, 148
Node instance probe
ACTIVE_INSTANCES_HIGH, 83
ACTIVE_INSTANCES_LOW, 83
ACTIVE_VALUES_HIGH, 84
ACTIVE_VALUES_LOW, 83
AVAILABILITY, 83
INSTANCE_TPUT_HIGH, 84
INSTANCE_TPUT_LOW, 84
RESPONSE_TIME, 83
SETUP_TIME, 83
TRANSFER_TPUT, 83
Username, 77
VALUE_TPUT_LOW, 84
Node instances
C_OVBPI_NODE - OVBPI_NODE probe, 76
Node instances probe
OVBPI_DB_Password, 77
OVBPI_DB_User_Name, 77
OVBPI_Flow_Namet, 78
OVBPI_Identifier, 77
OVBPI_Node_Name, 78
Port, 77
Target Host, 77
NodeName
Nodes table, 146

Nodes table, 146
ActiveCount, 146
ActiveWeight, 147
AvrgTime, 147
Flow_ID, 146
InstanceRate, 147
LastRateUpdate, 148
LastRateUpdateLongMillis, 148
Node_ID, 146
NodeDescription, 146
NodeName, 146
NodeType, 146
ResourceStatus, 148
SampleCount, 147
TotalCount, 146
TotalTime, 147
TotalWeight, 147
WeightRate, 147
X_Pos, 146
NodeType
Nodes table, 146
Notification Server
Administration Console, 41
architecture, 33
component of OVBPI Server, 33
email alerts and HP OpenView Service Desk, 36
OVBPI Server component, 22
Velocity email templates, 35
XSLT email templates, 35
NULL
when used within an expression, 187
Numeric
data type coercion rule, 184

## O

Objective information
C_OVBPI_FLOW_PROBE, 80
C_OVBPI_METRIC_PROBE, 82
C_OVBPI_NODE_PROBE, 83
for OVIS configuration, 79
openadaptor
event store, 161
handling OVBPI business events, 36
OpenView Dashboard
and OVBPI Dashboard, 57
OpenView Dashboard integration, 57
OpenView Service Desk
and Notification Server email alerts, 36
OpenView Service Desk processes
OVBPI monitoring, 53

OVBPI
Administration Console, 41, 42
and iWay, 56
and OVIS integration, 61
and OVSD, 89
and OVSD service mapping, 92
and SOA Manager integration, 97
architecture, 11, 45
architecture diagram, 12, 46
Business Event Handler, 36
custom probe configuration for multiple OVBPI Servers, 87
database
schema, 105
deployer, 41
importing SOA Manager business events, 98, 101
importing SOA Manager status events, 98
Modeler, 40
Model Repository, 40
monitoring OVSD processes, 53
Notification Server Administration Console, 41
openadaptor, 36
OVSD service calls, 90
Probe locations, 86
propagating events, 38
schema data type definitions, 107
schema entity relationship diagram, 156
Server components, 22
SLAs, 85
OVBPI_DB_Password
Flow Instances probe, 74
Flow Metric probe, 76
Node instances probe, 77
OVBPI_DB_User_Name
Flow Instances probe, 73
Flow Metric probe, 75
Node instances probe, 77
OVBPI_Flow_Name
Flow Instances probe, 74
Flow Metric probe, 76
Node instances probe, 78
OVBPI_Identifier
Flow Instances probe, 73
Flow Metric probe, 75
Node instances probe, 77
OVBPI_Metric_Name
Flow Metric probe, 76
OVBPI_Node_Name
Node instances probe, 78
OVBPI and OVIS
design-time integration, 63
run-time integration, 65
OVBPI data
used for developing applications, 106
OVBPI database
business process metric data, 20
diagram of relationship with OVBPI components, 19
email notification data, 21
event hospital data, 21
flow data, 20
metric threshold data, 21
model repository data, 21
part of architecture, 18
OVBPI Intervention Client, 42
OVBPI Modeler
and BPEL, 40
grammar description, 172
grammar rules, 171
OVBPI Notification Server Administration Console, 42
OVBPI OpenView Operations Adapter
architecture, 52
OVBPI Probes and alarms
architecture, 50

OVBPI Server
adapters, 26
architecture, 22
business object manager, 26
Business Process Metric definer component, 28
configuring custom probes, 87
event receivers and transmitters, 27
Metric Engine, 30
Notification Server, 33
OvbpiProbe.cfg, 87
OVIS event receiver, 27
Repository Explorer, 32
OVBPI SLO and SLA violations, 52
OVBPI SOA Manager
configuring access to adapter, 102
OV Dashboard
single signon, 58
OVIS
alarms and SLOs, 79
alarms database tables, 86
custom probes, 61, 68, 69
enabling integration, 86
event receiver for OVBPI Server, 27
integration with OVBPI, 61
Objective information, 79
Probes custom dialogs, 42
reporting on operational services, 61, 68
reporting SLO and SLA violations, 61, 68
OVIS and OVBPI
design-time integration, 63
integration, 63
run-time integration, 65
OVIS custom probe, 50
C_OVBPI_FLOW - OVBPI_FLOW probe, 72
C_OVBPI_METRIC - OVBPI_METRIC probe, 72
C_OVBPI_NODE - OVBPI_NODE probe, 72
Flow Instances probe, 72
Flow Metric probe, 74
Node instances probe, 76
OvpbiProbe.cfg
configuring for OVBPI, 87
OVSD
custom field definition, 92
integrating with OVBPI, 90
mapping services, 92
monitoring by OVBPI, 53
service calls, 90

## P

Patient
DBusMH_Patient table, 165
PatientId
DBusMH_Patient table, 163
PatientStatus
DBusMH_Patient table, 163
Port
Flow Instances probe, 73
Flow Metric probe, 75
Node instances probe, 77
Primary_entity
Flow_Instances table, 143
Flows table, 141
Primary_entity_ inst
Flow_Instances table, 143
Probe locations
for OVBPI custom probes, 86

Progression rule
after(), 178
before(), 178
case sensitivity for expressions, 182
changed(), 178
created(), 179
in(), 179
methods for, 178
terminated(), 179
Propagating events, 38
PubUniqueId
DBusMH_Patient table, 164

## Q

Quarter
Metric_Dim_Dates table, 122

## R

RaisedTime
Metric_Fact_Alerts table, 110
RejectReason
DBusMH_Patient table, 164
Reporting operational status using OVIS, 68
Reporting SLO and SLA violations using OVIS, 68
Repository Explorer
architecture, 32
component of OVBPI Server, 32
RepositoryID
Data_Objects table, 158
Flows table, 141
RepositoryRevision
Data_Objects table, 159
Flows table, 142
Resource_ID
Node2Resources, 154
Resources table, 153
ResourceDescription
Resources table, 153
ResourceName
Resources table, 153
Resources table, 152
DeploymentStatus, 154
LastChange, 154
LastChangeLongMillis, 154
Resource_ID, 153
ResourceDescription, 153
ResourceName, 153
ResourceType, 153
RootCause, 153
ResourceStatus
Node_Instance table, 149
Nodes table, 148
ResourceType
Resources table, 153
RESPONSE_TIME, 80, 82, 83
Retried
DBusMH_Patient table, 165
RootCause
Resources table, 153
Rules
for Modeler grammar, 171
for thresholds, 171

## S

SampleCount
Data_Objects table, 158
Flows table, 140
Nodes table, 147

Schema, 105
Arcs
table in schema, 152
Business Entity, 157
Business Entity Data Type, 157
Business Entity Keys, 157
Business Event Handler, 161
Business process metrics, 108
definition, 105
dimensional versus entity model, 108
Flow_Instances, 142
Flows, 139
Metric_Custom_Types, 134
Metric_Dim_Dates, 122
Metric_Dim_Flow_Instances, 126
Metric_Dim_Flows, 124
Metric_Dim_Groups, 128
Metric_Dim_Metrics, 121
Metric_Dim_Thresholds, 129
Metric_Fact_Alerts, 110
Metric_Fact_Statistics, 117
Metric_Fact_Values, 113
Metric_Generate_Errors, 135
Metric_Values, 137
Node2Resources
table, 154
Node_Instance_CompletedTimes, 151
Node_Instance_StartedTimes, 150
Node_Instance table, 148
Nodes table, 146
Resources table, 152

Security
architecture, 43
Select Access, 43
Tomcat Servlet Engine, 43

Select Access
security, 43

Self-Healing Services
architecture, 59

Service Desk
and Notification Server, 36
custom field definition, 92
integrating with OVBPI, 90
integration
through adapter, 89
integration through Business Process Dashboard, 89
mapping services, 92
monitoring by OVBPI, 53

Service Level Agreements
for OVBPI, 85

SETUP_TIME, 80, 82, 83

Single signon
for OV Dashboard links to OVBPI, 58

SLA
for OVBPI, 85

SLO and SLA violations
using OVIS, 52

SLO information, 80, 81, 82, 83, 84
C_OVBPI_FLOW_PROBE, 80
C_OVBPI_METRIC_PROBE, 82
C_OVBPI_NODE_PROBE, 83

SOA Manager
business events, 98, 101
configuring access to adapter, 102
integration with OVBPI, 97, 98
status of Business Services, 98

Source
Arcs table, 152

SourceAppName
DBusMH_Patient table, 164

StagingAlertLevel
Metric_Dim_Thresholds table, 133

Start and complete on transition
grammar rules, 176

StartCondition
Metrics table, 136

Start condition
after(), 178
before(), 178
changed(), 178
created(), 179
in(), 179
methods for, 178
terminated(), 179
StartedTime
Node_Instance_StartedTimes table, 150
StartedTimeLongmillis
Node_Instance_StartedTimes table, 150
StartNode
Metrics table, 136
StartTime
Data_Object-Instance table, 159
Flow_Instances table, 144
Metric_Dim_Flow_Instances table, 127
Metric_Fact_Value table, 115
Metric_Values table, 137
Node_Instance table, 149
StartTimeLongMillis
Data_Object-Instance table, 159
Flow_Instances table, 144
Metric_Values table, 137
Node_Instance table, 149
Statistics facts dimension, 116
StatisticsType
Metric_Fact_Statistics table, 118
STATUS
EVENT_STORE table, 162
Status
Data_Object-Instance table, 160
Data_Objects table, 158
Flow_Instances table, 145
Flows table, 141
Metric_Dim_Flow_Instances table, 127
Metric_Dim_Flows table, 125
Metric_Fact_Value table, 114
Node_Instance table, 149
Resources table
Resources table
Status, 153
StdDevValue
Metric_Fact_Statistics table, 119
String
data type coercion rule, 184
data type definition, 107
operator for coercion, 185
values for flow definitions, 174
Subclass, 145
Data_Objects table, 159
Flow_Instances table, 145
Flows table, 141
Subject
DBusMH_Patient table, 163
SumSquareValue
Metric_Fact_Statistics table, 119
SumValue
Metric_Fact_Statistics table, 118

## T


Table
- Arcs, 152
- Data_Object-Instance, 159
- EVENT_STORE, 161
- Flow_Instances, 142
- Flows, 140
- Metric_Custom_Types, 134
- Metric_Dim_Dates, 122
- Metric_Dim_Flow_Instances, 126
- Metric_Dim_Flows, 124
- Metric_Dim_Groups, 128
- Metric_Dim_Metrics, 121
- Metric_Dim_Thresholds, 129
- Metric_Fact_Alerts, 110
- Metric_Fact_Statistics, 117
- Metric_Fact_Values, 113
- Metric_Generate_Errors, 135
- Metric_Values, 137
- Node2Resources, 154
- Node_Instance_CompletedTimes, 151
- Node_Instance_StartedTimes, 150
- Nodes, 146
- Resources, 152

Target Host
- Flow Instances probe, 73
- Flow Metric probe, 75
- Node instances probe, 77

TBN_DURATION_HIGH, 82

TBN_DURATION_LOW, 82

Text
- data type definition, 107

Threshold
- grammar rules, 171

ThresholdColumnName
- Metric_Dim_Thresholds table, 130

Threshold definitions, 171

ThresholdDescription
- Metric_Dim_Thresholds table, 129

ThresholdMessage
- Metric_Dim_Thresholds table, 129

ThresholdName
- Metric_Dim_Thresholds table, 129

ThresholdTest
- Metric_Dim_Thresholds table, 131

ThresholdType
- Metric_Dim_Thresholds table, 130

Throughput
- Metric_Fact_Statistics table, 118

Time
- Metric_Dim_Dates table, 124
- Metric_Fact_Alerts table, 110
- Metric_Fact_Statistics table, 117
- Metric_Values table, 138

TIME_RX
- EVENT_STORE table, 162

TIME_TX
- EVENT_STORE table, 162

TimeCompleted
- Metric_Fact_Value table, 114

Tomcat Servlet Engine
- security, 43

TotalCount
- Nodes table, 146

TotalFlows
- Flows table, 140

TotalInstances
- Data_Objects table, 158

TotalTime
- Nodes table, 147

TotalWeight
- Nodes table, 147

TRANSFER_TPUT, 80, 82, 83

Treated
- DBusMH_Patient table, 164

TreatedUser
DBusMH_Patient table, 165
Type
Arcs table, 152
Metric_Values table, 137

## U

Username
Flow Instances probe, 73
Flow Metric probe, 75
Node instance probe, 77

## V

Value
Metric_Fact_Alerts table, 111
Metric_Fact_Value table, 115
Metric_Values table, 137
VALUE_TPUT_HIGH, 81
VALUE_TPUT_LOW, 81, 84
Value facts dimension, 112
ValueUnits
Metric_CustomTypes table, 134
Metric_Dim_Metrics table, 121
variable-length binary
data type definition, 107
variable-length text
data type definition, 107

## W

Week
Metric_Dim_Dates table, 123
Weight
Flow_Instances table, 143
Metric_Fact_Value table, 114
Weight_type
Flow_Instances table, 144
WeightAverageValue
Metric_Fact_Statistics table, 120
WeightRate
Nodes table, 147
WeightStdDevValue
Metric_Fact_Statistics table, 120
WeightSumSquareValue
Metric_Fact_Statistics table, 119
WeightSumValue
Metric_Fact_Statistics table, 119
WeightThroughput
Metric_Fact_Statistics table, 119
WeightTotal
Metric_Fact_Statistics table, 119
WeightType
Metric_Dim_Flow_Instances table, 126

## X

X_Pos
Nodes table, 146

## Y

Y_Pos, 146
Nodes table, 146
Year
Metric_Dim_Dates table, 122