HP OpenView Business Process Insight

System Administration Guide

Software Version: 02.00

January 2006

Legal Notices

Support

Please visit the HP OpenView web site at:

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

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

You can also go directly to the support web site at:

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

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 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

This guide provides information about the management tasks that you need to complete for the ongoing set up and maintenance of your OpenView Business Process Insight (OVBPI) solution.

This chapter describes the tools and procedures that are available for managing OVBPI. Some of these tools are provided with OVBPI and some are tools that are available with the third-party products that OVBPI uses, for example, database tools.

OVBPI comprises a number of different components, some of which can be distributed to other systems. For example, adaptors are typically located on the same system as the application that they are connecting to, and OVIS probes are installed on the system where OVIS is installed. There are some OVBPI components that always need to be located together and these are collectively referred to as the OVBPI Server components.

Chapter 2, OVBPI Architecture describes the architecture of the OVBPI system and how all these components are related. You do not need to read this chapter in order to manage the OVBPI system; however, it provides useful information that helps to understand how the components relate to each other.

The remainder of the guide describes the administration tools that are available and how to use these tools in the maintenance of your OVBPI system.

Tools Available to Manage OVBPI

The following are the tools that are available to you for managing your OVBPI system:

- OVBPI Server Administration Console

  This console enables you to modify parameters relating to the OVBPI Server components. The console, and details of the components that can be managed through it, are described in Chapter 3, OVBPI Component Administration.

- Log files

  Use the log files to identify what is happening in your OVBPI system. You can set the logging to report at different levels of detail when you are trying to identify specific problems. Refer to Chapter 3, OVBPI Component Administration for details of how to set the logging levels for OVBPI Server components through the Administration Console. Refer to Chapter 6, OVBPI Modeler Administration for details of the OVBPI Modeler log files.

- Notification Server Administration Console

  Use this console to add and delete users who want to receive email alerts and OVO SLO and SLA notifications from OVBPI. You can also use this console to configure scripts that the Notification Server runs when it receives a notification event; see Chapter 9, Notification Server Configuration.

- Engine Instance Cleaner

  Use the Business Impact Engine Instance Cleaner to remove active or completed Flow and Data instances from the database and the OVBPI system. Typically, you do not use the Engine Instance Cleaner to remove individual Flow or Data instances. To remove individual instances, use the Intervention Client. Both the Engine Instance Cleaner and the Intervention Client are described in Chapter 10, Intervention.

- Repository Explorer

  Use the Repository Explorer to provide you with a view of the Model Repository data. It enables you to:

  — view and print details of the definitions that are stored in the Repository.

  — export current and superseded versions of your business flows.

  — delete definitions.

  — view the history of your definitions.

- Intervention Client

  The Intervention Client enables you to intervene or gain access to instances of Flow and Data definitions. You might need to do this in cases where the flow instance is blocked for some reason and cannot progress, or where you have entered inaccurate start and complete conditions in your progression rules.

  The Intervention Client also enables you to change the status of Services defined for your business flows, delete instances and undeploy definitions. Refer to Chapter 10, Intervention for details of the Intervention Client

  If you want to delete multiple Flow and Data instances that are active or complete, on a regular basis, use the Engine Instance Cleaner.

- Integrity Checker

  The Installation Integrity Checker checks the OVBPI installation status and compares the current status of your OVBPI system to its status immediately following its installation. For example, it checks file permissions and the J2SE version. The Integrity Checker is described as one of the problem solving tools in the *OpenView Business Process Insight Problem Solving Guide*.

Common Management Tasks

The following are examples of some of the management tasks that you might need to complete and details of where you can find more information about them in this guide:

- Changing logging levels for components.

  If you are trying to gather information about a particular problem that is occurring in your OVBPI system, or you are trying to debug a Flow definition that you have created, you might need to modify the level at which the OVBPI components are logging. Chapter 3, OVBPI Component Administration describes how to change the logging levels for OVBPI components.

- Configuration of the OVBPI system.

  Use the Administration Console to modify the OVBPI configuration parameters as described in Chapter 3, OVBPI Component Administration.

- Configuring your connectivity to HP OpenView Operations (OVO).

  You can configure an OVBPI Server to receive service impact information from OVO either on HP-UX through HP OpenView Service Navigator (OVSN), or on Windows using HP OpenView Operations for Windows (OVOW). (An OVBPI Server cannot connect to both OVSN and OVOW at the same time). You need to configure your OVBPI Server using the Administration Console if you want to set this configuration.

- Configuring connectivity to OVIS

  You can configure an OVBPI Server to receive service impact information from OVIS. You need to configure your OVBPI Server using the Administration Console if you want to set this configuration.

  Configuring OVBPI to receive service impact information is separate from configuring OVBPI customized probes for OVIS.

- Configuring OVIS probes for OVBPI flows that you have installed.

  Installing OVIS probes is an optional task, but 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. Installing the probes is described in the *OpenView Business Process Insight Installation Guide*.

- Configuring OVBPI to be able to access HP OpenView Service Desk incident reports.

  If HP OpenView Service Desk (OVSD) is a component in your system, you can configure OVBPI to link to OVSD incident reports and display their details through the Business Process Dashboard; see Chapter 5, OVBPI and OVSD.

- Managing the OVBPI Modeler.

  There are some management tasks associated with the OVBPI Modeler, for example, importing and exporting flow definitions. These tasks are described in Chapter 6, OVBPI Modeler Administration.

- Managing the Business Process Metric definer.

  There are some management tasks associated with the Business Process Metric Definer, which are described in this guide, for example, exporting and importing metric definitions. These tasks are described in Chapter 7, OVBPI Metric Definer Administration.

- Modifying the Business Process Dashboard.

  You can modify the code for the Business Process Dashboard in order to integrate the OVBPI Dashboard into your solution, for example, you might have a Web portal that you want to use to show the impact information for your business flows. Modifying the example Business Process Dashboard is described in *HP OpenView Business Process Insight Integration Training Guide - Customizing Dashboards*.

- Managing business flow data in the Model Repository,

  You can export definitions from the Business Impact Engine using the Repository Explorer. You can also use the Repository Explorer to view and print your flow, data and event definitions. These definitions are provided in a forms-style interface using the Explorer, which makes them easier to read and print.

  The Repository Explorer is described in Chapter 8, Repository Explorer.

- Configuring user accounts to receive email alerts for SLO and SLA violations, flow event notifications and metric threshold alerts.

  You use the Notification Server to configure user accounts to subscribe to event notifications and alerts. You can also subscribe user accounts to events, such that when a specified subscription event is received by the Notification Server, a script is run.

  Using the Notification Server Web Administration Console is described in Chapter 9, Notification Server Configuration.

- Intervening in flow progression.

  The following are examples of why you might need to intervene in a flow:

  — where you have Flows that are not progressing, perhaps because someone has manually completed a step in the flow, when a service is not available

  — there is an error in the start and complete conditions.

  Chapter 10, Intervention describes the tools that are available to intervene in the progression of a flow.

- Database housekeeping.

  If you are monitoring many OVBPI flow instances, the data in the OVBPI database can build up over time. Chapter 10, Intervention describes the tools and parameters that are available to manage the data in the OVBPI database.

- Backup and recovery.

  For production systems, you need to make sure that you have a backup policy for your OVBPI system that fits in with your overall organizational backup policy. Chapter 11, Backup and Recovery describes how to backup OVBPI and alerts you to any specific requirements that might have an effect on your organizational backups.

- Problem solving.

  If you have specific problems with your system that you need to resolve, refer to the *OpenView Business Process Insight Problem Solving Guide*, as the problem and recovery might be documented. If you are unable to solve a particular problem, report it to HP, and include the information requested in the *OpenView Business Process Insight Installation Guide*.

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:

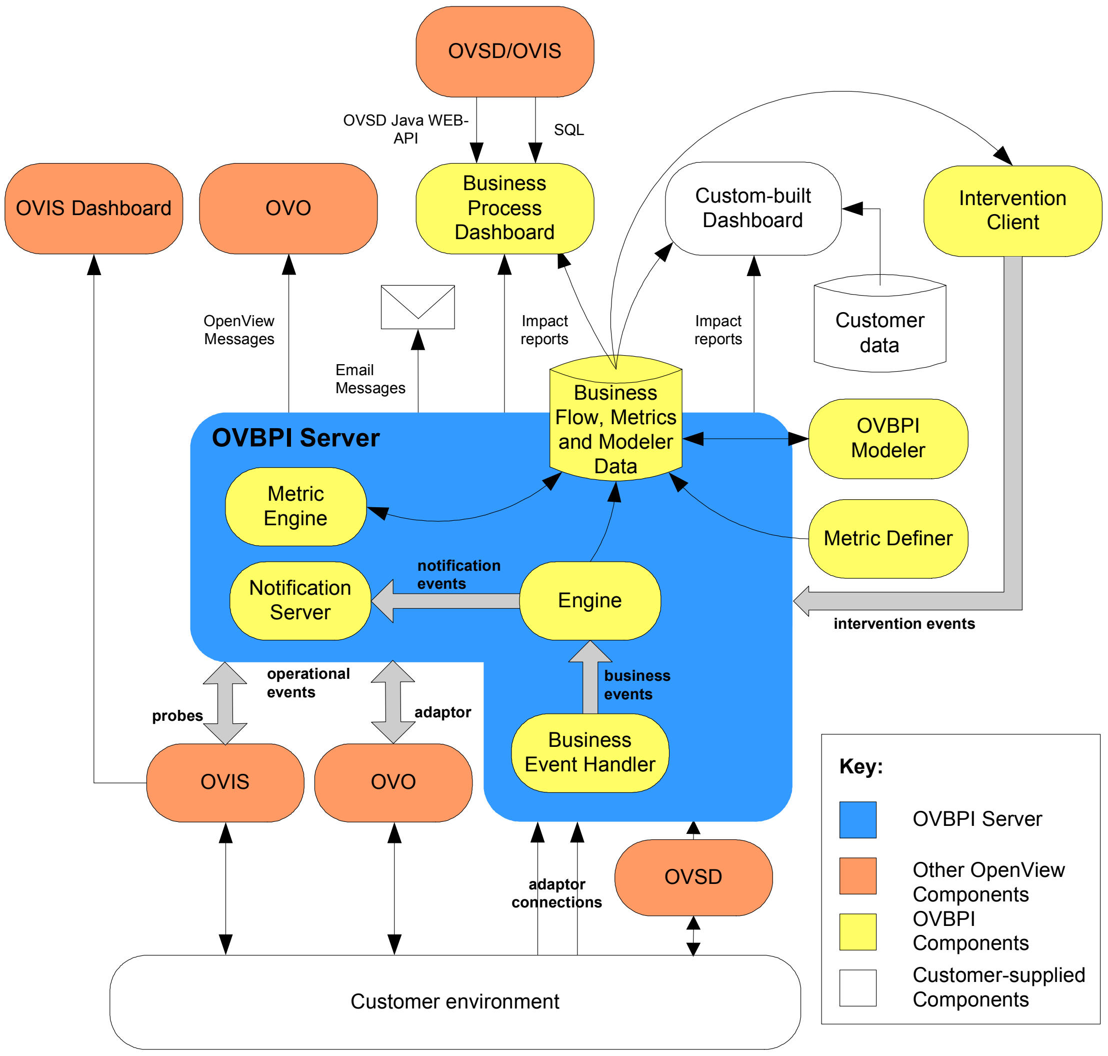
- A high-level representation of the OVBPI component architecture; see Figure 1 on page 21.
- The OVBPI Business Process Dashboard, which is used to display the impact information from the OVBPI system; see section Business Process Dashboard on page 22.
- The OVBPI Intervention Client, which provides you with access to active business flows and flow data; see section Intervention Client on page 25.
- The OVBPI database and how it is used by the OVBPI components; see section OVBPI Database on page 27
- The OVBPI Server components; see section OVBPI Server on page 32.

- The OVBPI Modeler and the data used and stored by the OVBPI Modeler; see section Modeler on page 55.
- The Metrics definer, which you use to define business process metrics and metric thresholds for your flows; see section Business Process Metric Data on page 29 and section Metric Threshold Definitions on page 30.
- The administration GUIs and consoles used within the OVBPI system; see section Administration Consoles and Interfaces on page 57.

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.

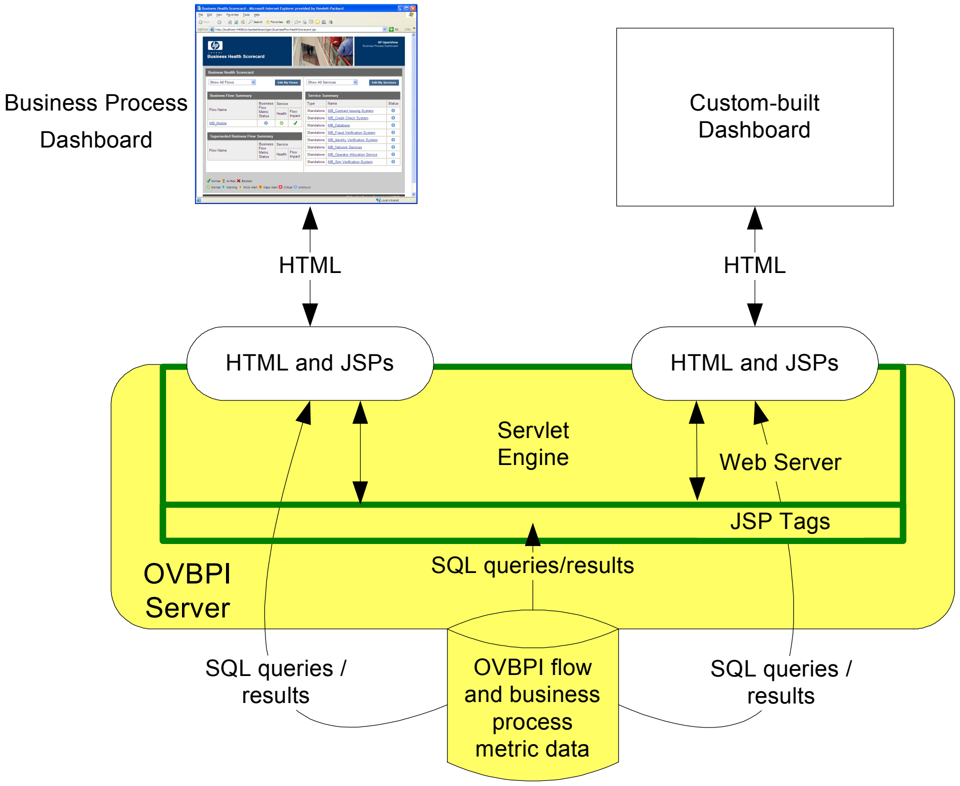
The OVBPI Dashboard also enables you to associate the service information received from OpenView Operations (OVO) and OpenView Internet Services (OVIS) with OpenView Service Desk (OVSD) service calls and incidents; see Chapter 5, OVBPI and OVSD.

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 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.

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 24). 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 *OVBPI Integration Training Guide - Customizing Dashboards* 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 *OVBPI Integration Training Guide - Customizing Dashboards* 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 Concepts and Modeling Flows*.

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 42). 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

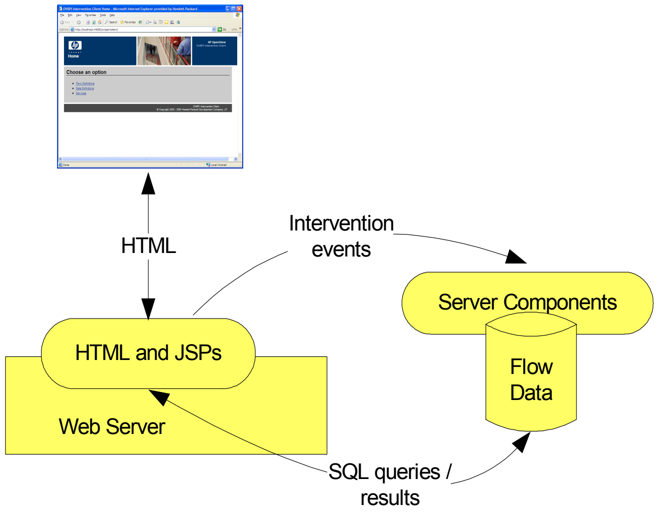
The Intervention Client enables you to access flows that you have deployed in order to modify or delete Flow instances and their associated Data instances. You might need to do this to resolve problems with the flow, or its data, usually after a period of new development or where progression rules are 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 26. 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 section Intervention Client on page 251.

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.

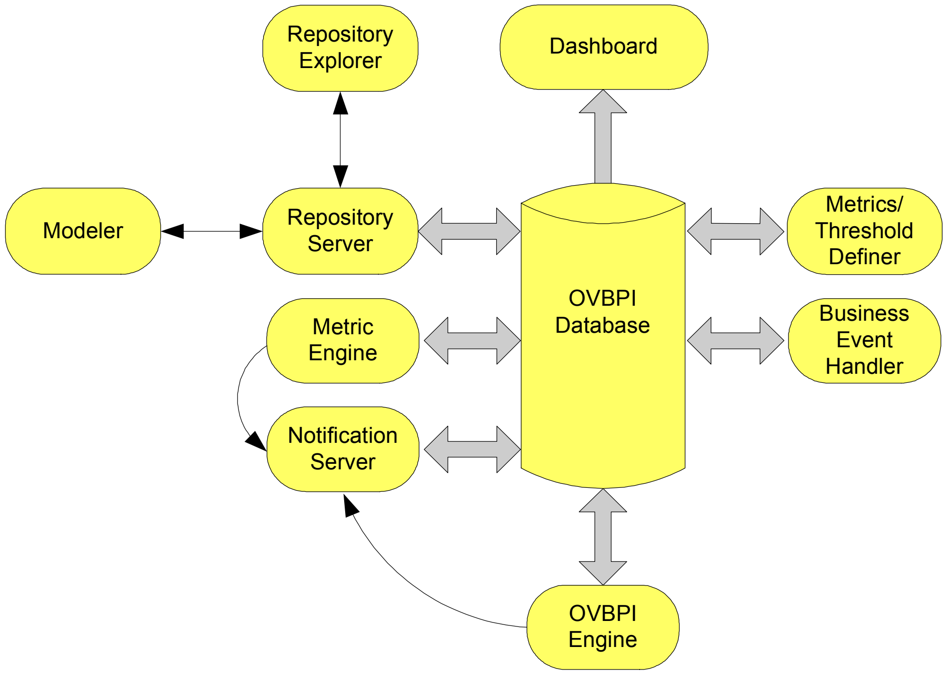
The database holds all the information that OVBPI needs to operate, with the exception of a number of configuration files, which are located under the OVBPI installation directory.

You can access some of 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 Appendix A, Database Schemas. 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 accessed or modified.

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 the OVBPI Server, or on a system that is remote from the OVBPI Server. 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 Concepts and Modeling Flows* and 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 Concepts and Modeling Flows*. Building a customized Dashboard is described in the *OpenView Business Process Insight Integration Guide - Customizing Dashboards*.

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 Concepts and Modeling Flows* and 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 browse, print, and a level of management for the Model Repository data.
- Monitoring business events, through adaptors, 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 business 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:

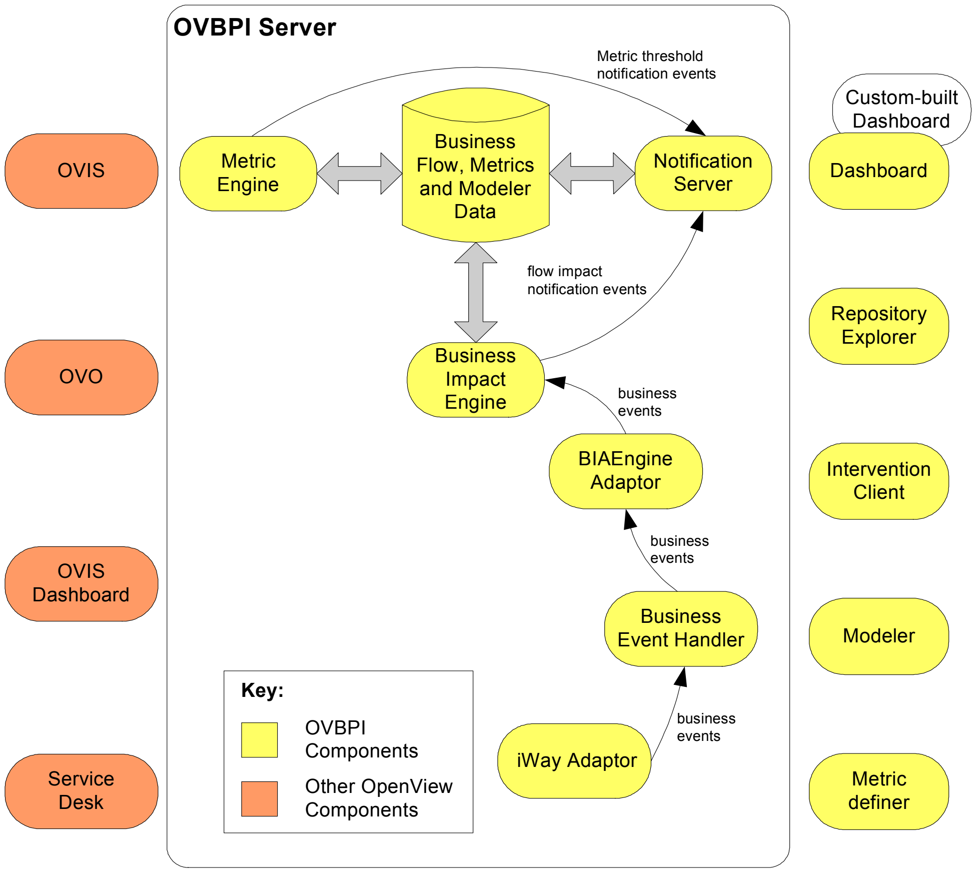
- Business Impact Engine
- Metric Engine
- Business Process Metrics definer
- Repository Explorer
- Notification Server

- Business Event Handler
- OVBPI OVIS service impact alarm and violation polling functions
- OVBPI OpenView Operations Adaptor

The OVBPI custom probes for OVIS are not part of the OVBPI Server; they are installed on the system where the OVIS Management Server is also installed; see section OVBPI and OVIS Probes and Alarms on page 49 for more information about the OVIS integration.

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

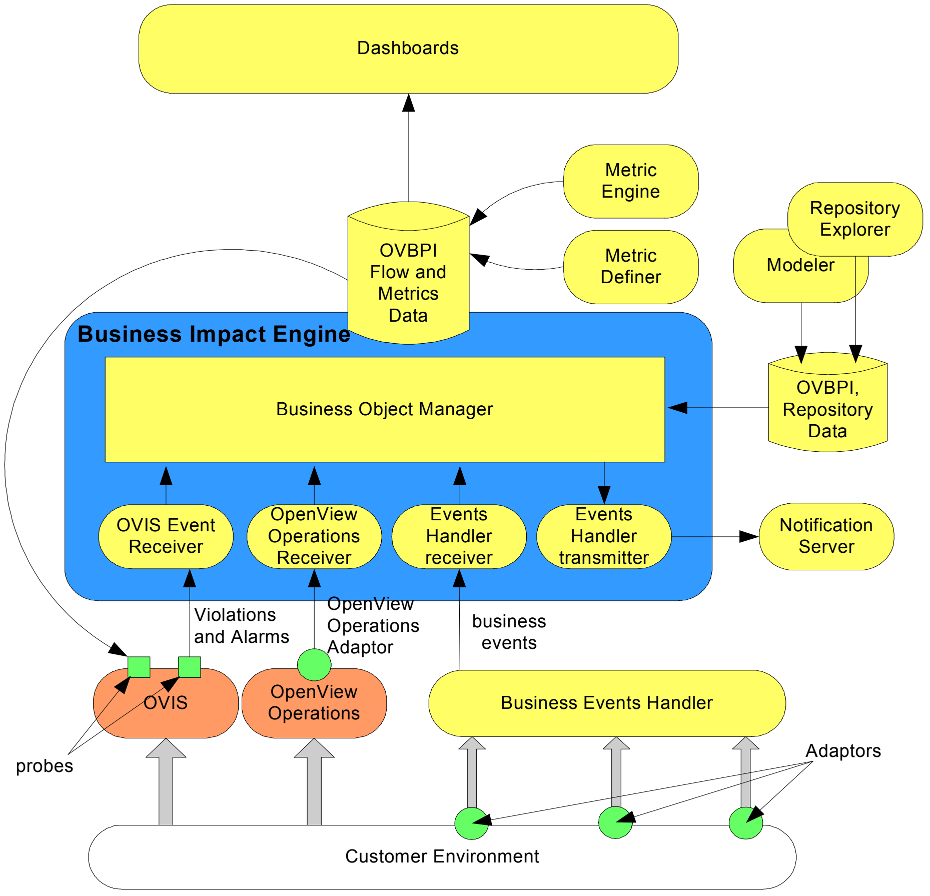


Business Impact Engine

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 38.

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 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 Adaptor

  The OVBPI OpenView Operations Receiver accepts operational events from the OpenView Operations Adaptor, 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 Adaptor 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.

- 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.

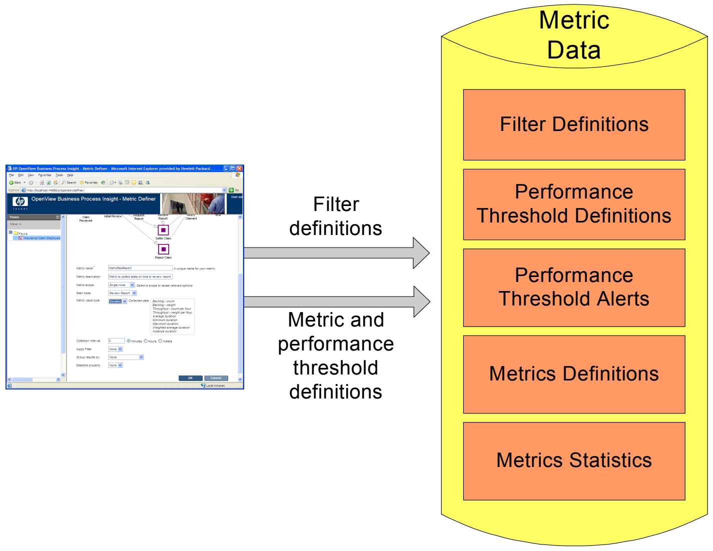
A business process metric is a business measurement that has a specific meaning within your business and can be used to record information about the business flow that you are monitoring. Statistics, such as average and maximum, can be calculated using metric instance data. In addition, metric thresholds 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


Metric Data
Filter Definitions
Performance Threshold Definitions
Performance Threshold Alerts
Metrics Definitions
Metrics Statistics
Filter definitions
Metric and performance threshold definitions
OpenView Business Process Insight - 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.

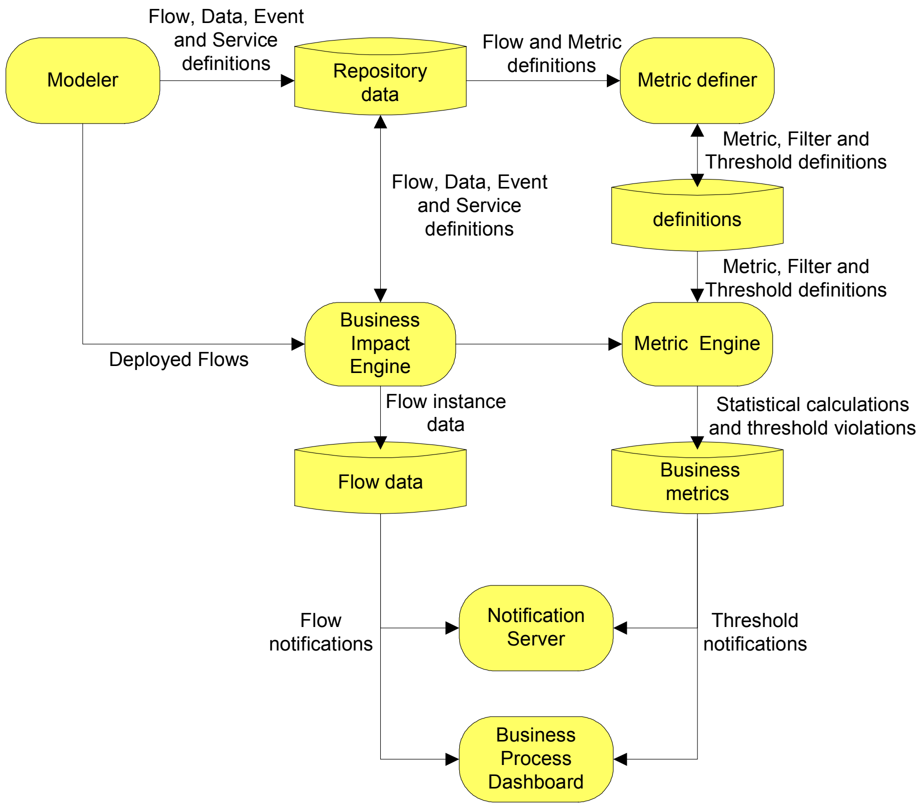
How the Business Process Metrics definer interworks with the Metric Engine is described in section Metric Engine on page 38.

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


Flow, Data, Event and Service definitions
Modeler
Repository data
Flow and Metric definitions
Metric definer
Metric, Filter and Threshold definitions
definitions
Flow, Data, Event and Service definitions
Metric, Filter and Threshold definitions
Deployed Flows
Business Impact Engine
Metric Engine
Flow instance data
Statistical calculations and threshold violations
Flow data
Business metrics
Flow notifications
Notification Server
Threshold notifications
Business Process Dashboard

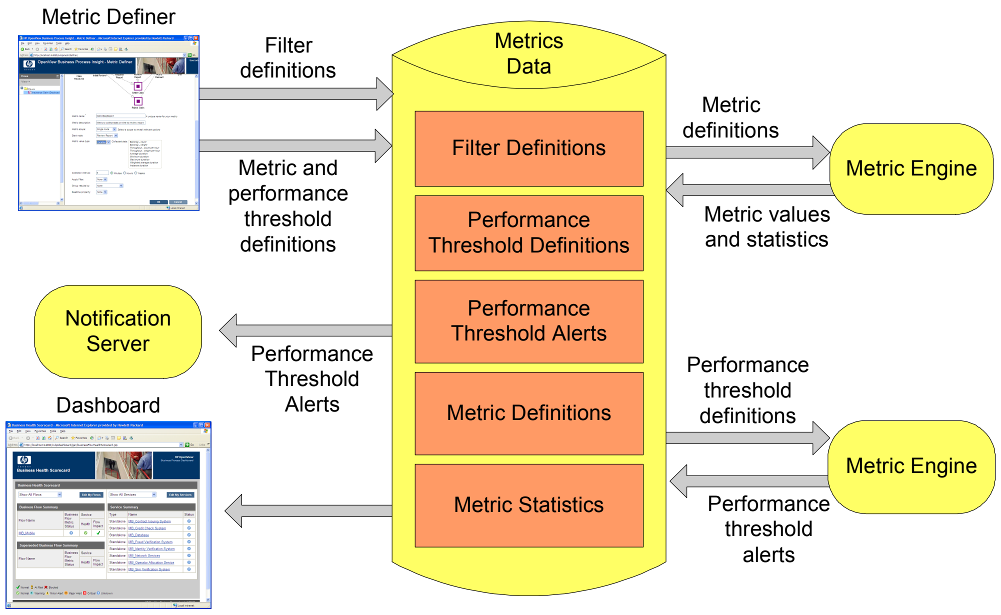
The business process metrics and metric thresholds are configured through the Business Process Metrics definer; see section Business Process Metric Definer on page 36.

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


Metric Definer
Filter definitions
Metric and performance threshold definitions
Metrics Data
Filter Definitions
Performance Threshold Definitions
Performance Threshold Alerts
Metric Definitions
Metric Statistics
Metric definitions
Metric Engine
Metric values and statistics
Notification Server
Performance Threshold Alerts
Dashboard
Performance threshold definitions
Metric Engine
Performance threshold alerts

These 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.

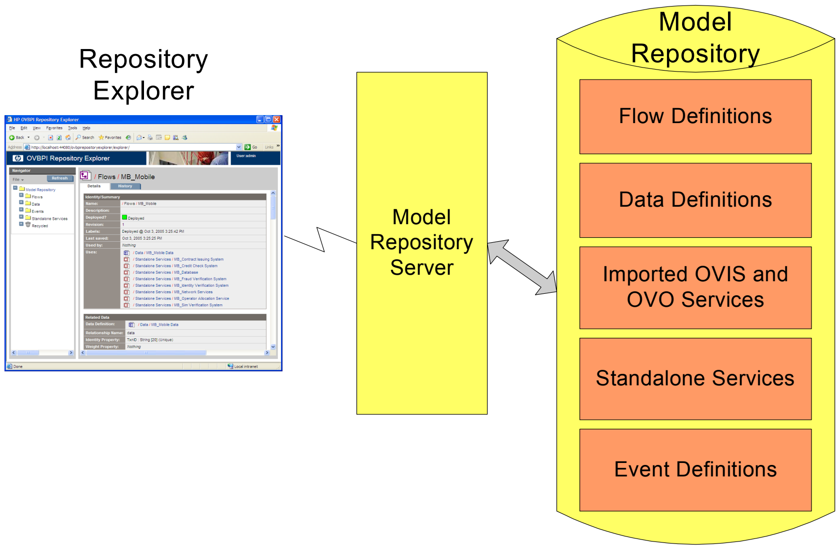
The Repository Explorer also provides access to information about all the superseded versions of the business flow that are currently deployed, enabling you to view and export details of the superseded flows.

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



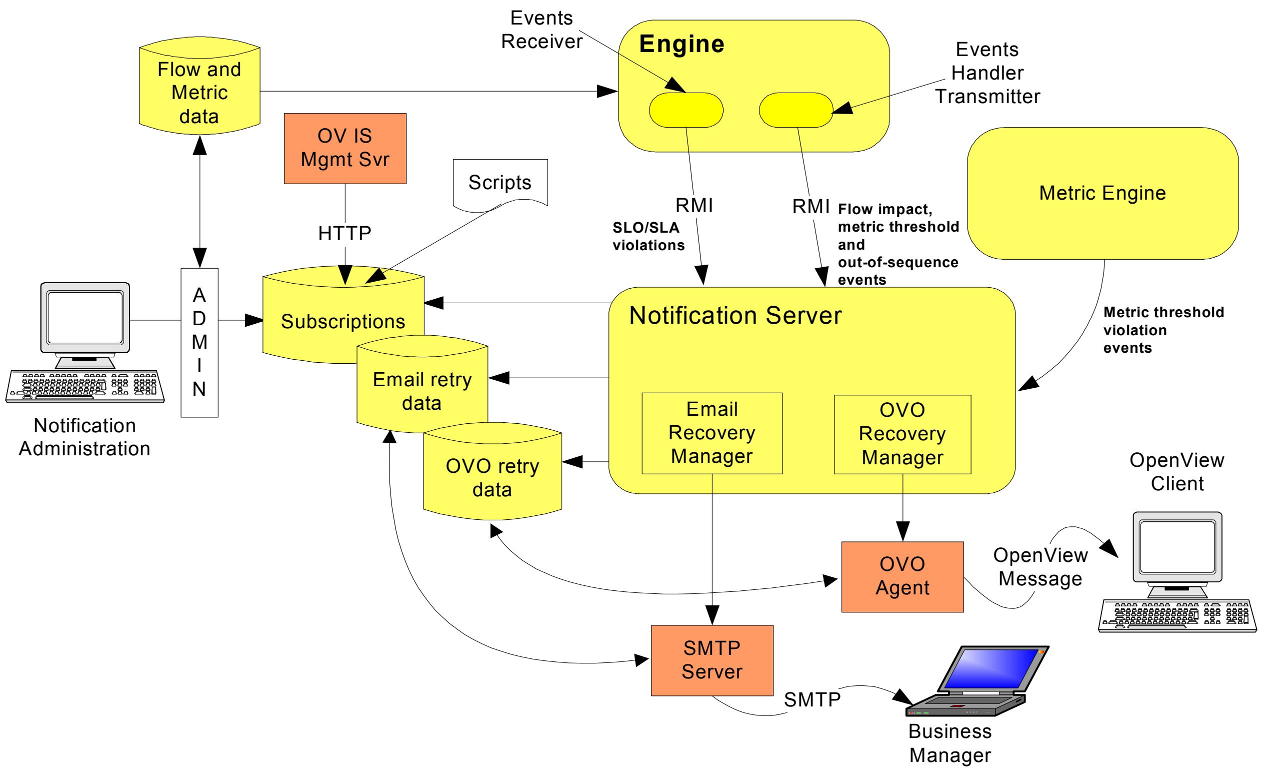
Notification Server

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 57.

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 Creating Scripts on page 241.

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. The default templates are provided as part of the OVBPI installation and instructions for creating your own templates are provided in section Creating Notification Server Templates on page 225.

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.

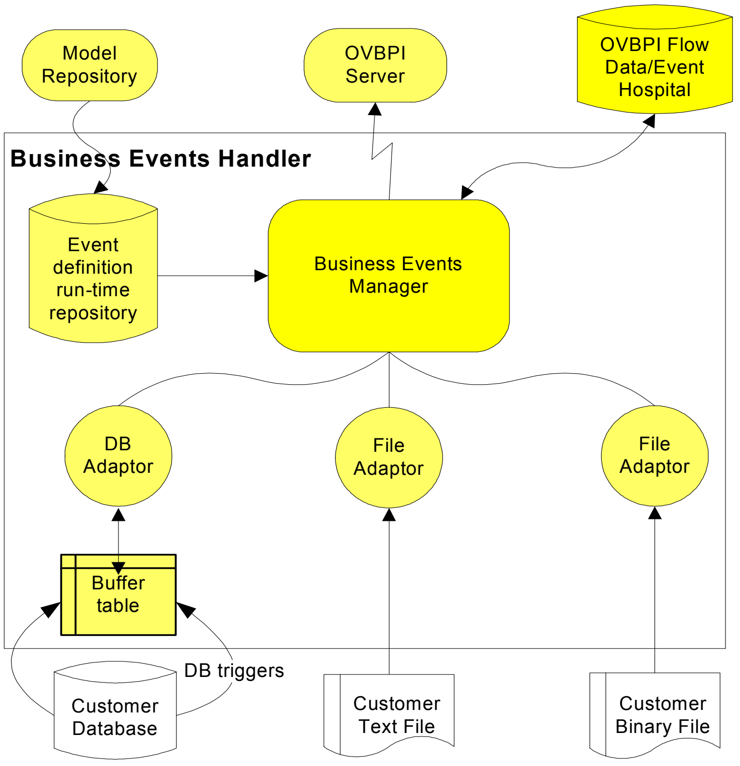
Business Event Handler

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 Events Manager 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 Events Manager also manages the business events that cannot be immediately delivered to the Business Impact Engine; for example, due to network failures.

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. The event retry interval depends on how the individual event adaptors for the source of the event data are configured; 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.

Where possible, events are automatically discarded from the Event Hospital. Events that have errors need to be repaired and manually discharged from the Event Hospital in order that they can be sent into the Engine.

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 `BIAEngineAdaptor` is notified of the change through a business event and adds the event details onto the event as defined in the OVBPI Modeler. (The `BIAEngineAdaptor` is the 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 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 attempts to send the event at a later time.

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 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 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.

OVBPI and OVIS Probes and Alarms

OVIS is not a component of OVBPI; however it can be an important part of the OVBPI solution. OVIS provides OVBPI users with information on Internet-related services. Specifically, it can be used to predict, isolate, diagnose and troubleshoot problems on the services that it is configured to monitor.

The OVBPI Modeler recognizes OVIS Services and imports these services into the Modeler, where you can associate them 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.

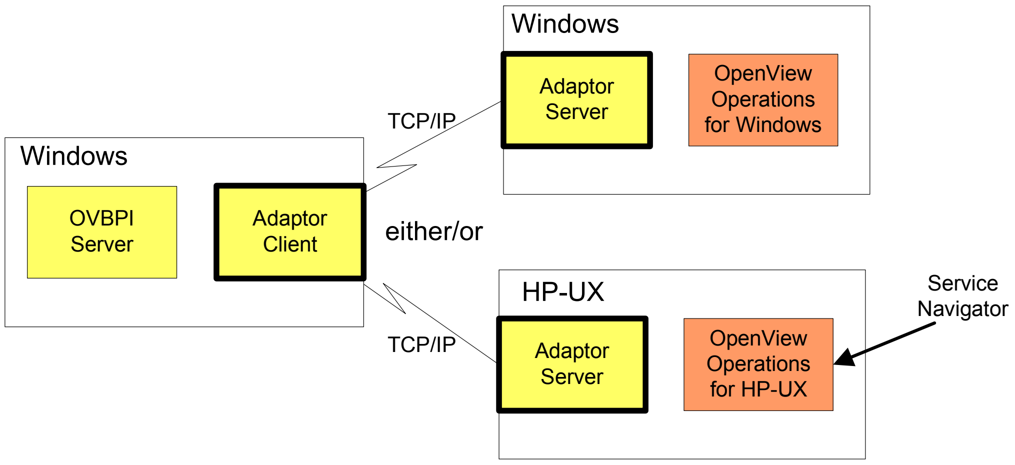
OVBPI OpenView Operations Adaptor

As in the case of OVIS, OpenView Operations is not a component of OVBPI; however, it can be an important part of the OVBPI solution. OVBPI uses OpenView Operations to report on the status of operational services that it has been configured to monitor.

The OVBPI OpenView Operations Adaptor 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 13 OVBPI OpenView Operations Adaptor



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 adaptor 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 Adaptor.

There are two components that make up the adaptor:

- The Adaptor Client component (referenced as the OpenView Operations Receiver in Figure 6 on page 34), which is installed with the OVBPI Server.
- The Adaptor 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 Adaptor for OVBPI are provided in the *OpenView Business Process Insight Installation Guide*.

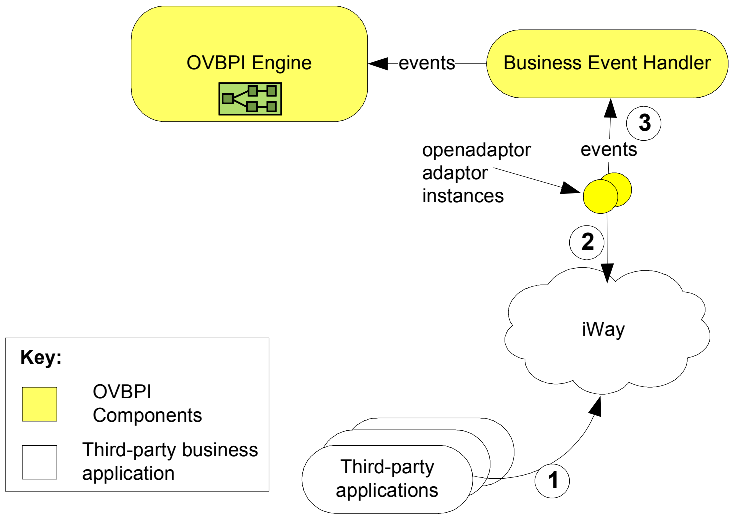
iWay Adaptor

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 adaptors, particularly for integrating well-known business applications, such as SAP, PeopleSoft and JD Edwards.

The OVBPI integration with iWay is achieved through an iWay adaptor for openadaptor, as shown in Figure 14.

Figure 14 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*.

HP OpenView Service Desk Adaptors

OVBPI provides a number of OVSD adaptors that you can use with OVBPI to monitor OVSD ITIL processes.

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

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

These adaptors are part of OVBPI and can be used with an OVBPI license or with the HP OpenView Service Desk Process Insight license bundle for OVBPI, which is provided specifically for OVSD integration with OVBPI.

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

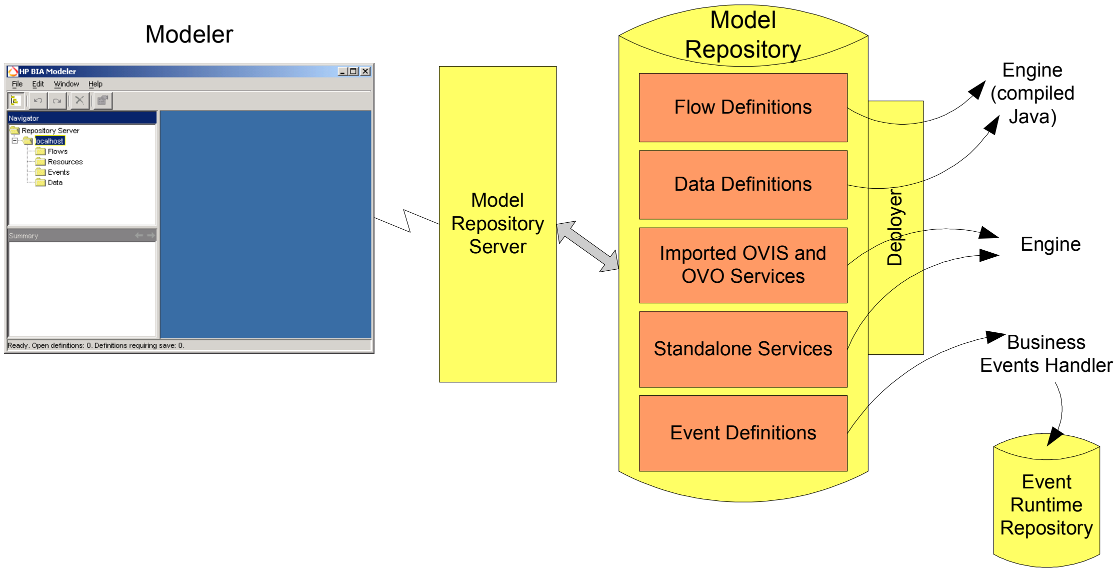
Modeler

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.

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.

Figure 15 OVBPI Modeler



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 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 Concepts and 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 Chapter 3, OVBPI Component Administration.
- 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 Chapter 4, OVBPI and OVIS.
- 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 Chapter 9, Notification Server Configuration.
- 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 Chapter 10, Intervention.

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

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

OVBPI Component Administration

This chapter describes the OVBPI Administration Console and the parameters that can be modified through the console. Later chapters in this guide provide more information about the management tasks that you can complete using the Administration Console, specifically, describing the use of the console within the context of the task that you are trying to complete.

This chapter includes the following information:

- The parameters that can be modified using the Administrative Console.
- How to make sure that any modifications made through the Administrative Console are preserved.
- Accessing the License Manager software.

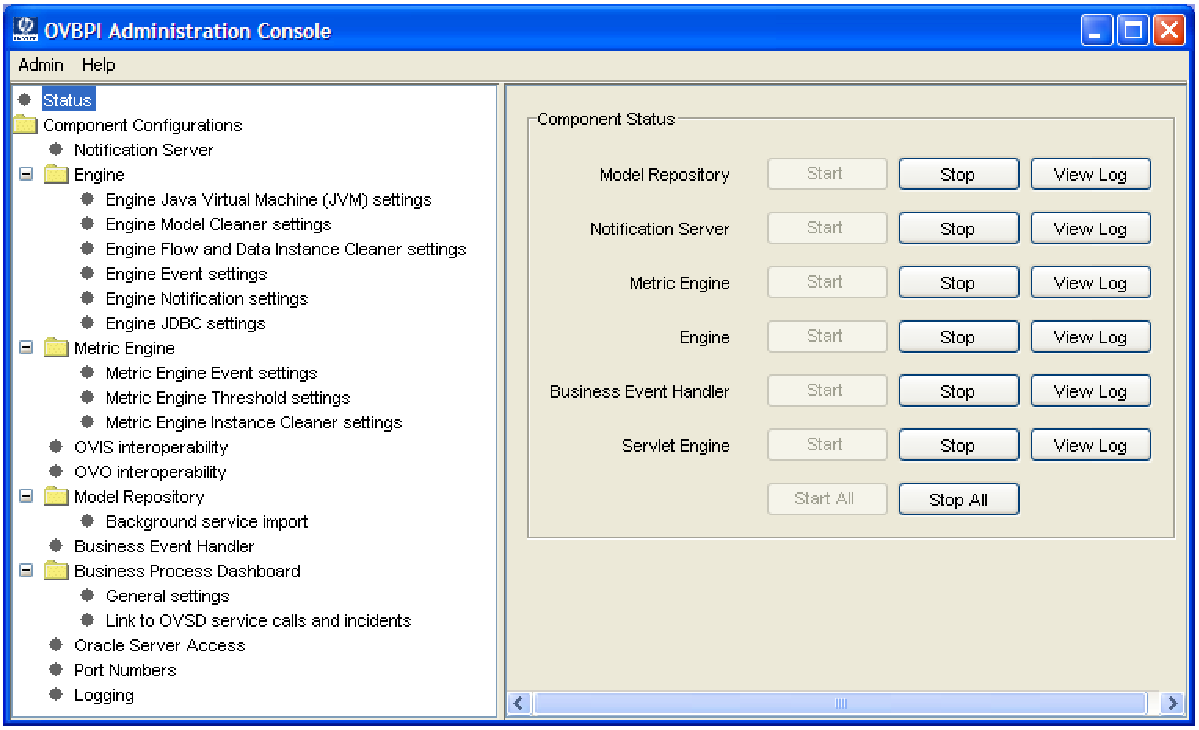
Not all the parameters described in this chapter need to be modified. What you need to change depends on your particular configuration and requirements. You are advised not to change anything unless you are recommended to in the documentation, or by your support representative.

Administration Console Description

You use the Administration Console to manage the OVBPI Server components. The OVBPI components that make up the OVBPI Server are described in the Chapter 2, OVBPI Architecture. The Administration Console is a graphical management interface, which displays a list of the parameters that can be modified. The Administration Console runs on both Microsoft Windows and HP-UX.

Figure 16 on page 60 shows an example of the Administration Console when it is first started on Windows following a full installation with Oracle as the configured database server.

Figure 16 Layout of Administration Console



The Administration Console opens on the `Status` panel, which is where you start and stop the OVBPI Server components.

You navigate through the Administration Console configurations using the explorer-style menu in the left-hand pane. Select the option from the pane that you want to manage. For example, if you want to manage port numbers select the `Port Numbers` option.

When you select an option in the left-hand pane, the right hand pane is updated to contain the configuration details for the parameters that can be modified. The console presents only the parameters for the components that are installed on the system where it is running. For example, on HP-UX, the console displays information about the OVBPI OpenView Operations Adaptor, port numbers and logging parameters.

You do not necessarily need to complete all the tasks described in this chapter for your implementation. Many of the parameters that you can change are used to configure more than one OVBPI component, for example, changing the database password affects all components that access the database.

➤ Make sure that you follow the instructions in each section when making changes to the configuration for OVBPI Server components through the Administration Console. This is because the steps for changing the parameters can vary according to the parameter that is being modified.

Starting the Administration Console

You start the Administration Console as follows:

- On Windows, select:

  `Start|Programs|HP OpenView|Business Process Insight|OVBPI Admin`

- On HP-UX, type the following command:

  ```
  OVBPI-install-dir/bin/biaadmin.sh
  ```

The Administration Console opens at the `Status` pane.

You can close the Administration Console at any time without applying outstanding changes and it does not impact the status of the system. The next time that you start the Administration Console, it shows the current status of the OVBPI components.

Functions Provided by the Administrative Console

OVBPI comprises a number of individual components, each of which has its own configuration, or property, file that define the parameters that can be used to control how the component behaves. Examples of parameter values that can be configured are:

- retry delays
- port numbers
- logging levels

Many of the parameters in these property files do not need to be changed. Therefore, in order to simplify the management of the OVBPI system, only those parameters that do need to be modified have been made available through the OVBPI Administration Console.

Specifically, the Administration Console provides access to the following OVBPI Server component areas:

- Server component status, which shows whether or not the OVBPI Server components are running and enables you to start and stop components; see section Status on page 64.
- Notification Server parameters; see section Component Configurations - Notification Server on page 76.
- Business Impact Engine parameters; see section Component Configurations - Engine on page 80.
- Metric Engine parameters; see section Component Configurations - Metric Engine on page 94.
- OVIS parameters; see section Component Configurations - OVIS Interoperability on page 102.
- OpenView Operations Adaptor parameters; see section Component Configurations - OVO Interoperability on page 106.
- OpenView Service Desk integration with the Dashboard parameters; see section Component Configurations - Model Repository on page 110
- Model Repository parameters; see section Component Configurations - Model Repository on page 110.
- Business Event Handler parameters; see section Component Configurations - Business Event Handler on page 113.

- Business Process Dashboard parameters; see section Component Configurations - Business Process Dashboard on page 116.
- Microsoft SQL Server access; see section Component Configurations - MS SQL Server Access on page 122.
- Oracle Server access; see section Component Configurations - Oracle Server Access on page 124.
- Port number settings; see section Component Configurations - Port Numbers on page 126.
- Logging parameter settings; see section Component Configurations - Logging on page 131.

The configuration parameters that you can modify for these component areas are described in the remaining sections of this chapter.

➤ The Modeler is not an OVBPI Server component; its configuration is described in Chapter 6, OVBPI Modeler Administration.

In addition to individual component configurations, the Administration console provides access to the HP OpenView License Manager software; see section License Management on page 134.

Status

Use this option to start and stop the OVBPI Server components and also to view the component log files of these components.

The Status pane lists the OVBPI Server components that you can start and stop, and includes buttons labeled, `Start`, `Stop` and `View Log`. The components included on the Status pane, following a full installation on Windows, are:

- Model Repository

  This is the design-time repository where the OVBPI Modeler stores its definitions and in conjunction with the design-time Model Repository stores and deploys the business flows created using the OVBPI Modeler.

- Notification Server

  The component used to send email notifications of business events to configured recipients.

- Metric Engine

  The component that analyzes and provides the statistical results from business process metrics and performance indicators. The business process metrics are created using the Business Process Metrics definer.

- Engine

  The component responsible for managing the progress of flow instances, based on the events received from external sources.

- OpenView Operations Adaptor

  Used to manage the communication between the Business Impact Engine and OpenView Operations for Windows.

- Business Event Handler

  Used to manage the communication between the Business Impact Engine and other business applications, for example, databases and files.

- Servlet Engine

  This is the Web Server component (Tomcat) that is installed as part of OVBPI. This component manages the server-side of all the OVBPI administration consoles and dashboard. These are the Business Process Dashboard, Notification Server Web Administration Console and Intervention Client.

In addition, there are the `Start All` and `Stop All` buttons, which you can use to start and stop all OVBPI components.

➤ The status pane lists only those components that you have installed, for example, the OpenView Operations Adaptor option is available following an installation of the adaptor on HP-UX.

The section Starting and Stopping the OVBPI Server Components on page 66 describes the procedure for starting and stopping the OVBPI server components.

You also access the log files for the OVBPI Server components from the `Status` pane. This enables you to identify status and error information for that component. The level of detail shown in the log files is controlled by the logging levels that you set. Setting log levels is described in section Component Configurations - Logging on page 131.

If the log files are not displaying correctly on your system, you can change the viewer used to display the log files as described in section Log File Locations and Changing the Log Viewer on page 75.

Starting and Stopping the OVBPI Server Components

You can use the OVBPI Administration Console to start and stop the OVBPI Server components that you have installed on the system where the Administration Console is running.

The OVBPI Server components are also defined as Windows Services and as installed are started automatically when your machine starts. This chapter describes using the Administration Console to start and stop the OVBPI Server components; refer to the *OpenView Business Process Insight Installation Guide* for details of the Windows Services for the OVBPI Server components.

Starting the OVBPI Server Components Using the Administration Console

You start the OVBPI Server components from the `Status` panel. Click the `Start All` button to start all the components installed on the system. Alternatively, click the `Start` button adjacent to the individual OVBPI components that you want to start.

➤ There are some dependencies between components, for example, some components require the database to be started before they can start. You are therefore advised to use the `Start All` option to start the components unless instructed otherwise.

Stopping the OVBPI Server Components Using the Administration Console

Before stopping the OVBPI Server components, make sure that you have stopped the other OVBPI components that might rely on them, for example the:

- OVBPI Modeler
- Repository Explorer
- Business Process Dashboard
- Notification Server Administration Console
- Intervention Client

You stop the OVBPI components in the same way that you started them, from the `Status` panel of the Administration Console. Select the `Stop All` option to stop the OVBPI Server, or stop the components individually.

Note that stopping the individual components does not stop all the OVBPI components. There are some background components, not listed, which are also automatically started and stopped through the OVBPI Administration Console. These are components that need to be running before the listed OVBPI components can be started and stopped. These components are started when the Administration Console is started, or in the case of the Windows Services, when the individual Services are started.

Starting and Stopping OVBPI Web Clients

OVBPI has the following Web clients:

- Metric Definer

  This is a Web application that you use to configure business process metrics and metric threshold definitions. Using the Metric definer to configure business process metrics and thresholds is described in *OpenView Business Process Insight Concepts and Modeling Flows*. Opening the Metric definer is described in section Accessing the OVBPI Metric Definer on page 69.

- Notification Server Administration Console

  This is a Web application that you use to add and configure users who want to receive email alerts from OVBPI. You also use it to configure users to receive OpenView Operations messages. Opening the Notification Server Web Administration Console is described in section Accessing the OVBPI Notification Server Administration Console on page 72. Using the Notification Server Web Administration Console to configure the Notification Server users is described in Chapter 9, Notification Server Configuration.

- The Business Process Dashboard

  This is a Web application where the Web pages for monitoring the business flows are displayed. The Dashboard can be on any system that has access to the OVBPI server. You can load the Business Process Dashboard pages using a Web browser client through a URL as described in section Accessing the Business Process Dashboard on page 73.

- Repository Explorer

  This is a Web application that you use to manage entries in the Model Repository, which holds the details of the business flows that you have created using the OVBPI Modeler. The Repository Explorer is described in Chapter 8, Repository Explorer. Opening the Repository Explorer is described in Accessing the Repository Explorer on page 70.

- The Intervention Client

  This is a Web application that enables you to make changes to definitions and instances that are active in the Business Impact Engine. Starting and stopping the Intervention Client is described in section Accessing the Intervention Client on page 74.

The Web Server that manages the Web pages for the Web-based consoles, dashboard and clients is Tomcat. You start Tomcat using the `Servlet Engine` component on the OVBPI Administration Console. Be aware that starting and stopping the Servlet Engine has an impact on all the Web applications that use it.

Accessing the OVBPI Metric Definer

Before trying to start the OVBPI Metric definer, make sure the `Servlet Engine` is started. If it is not, you cannot access the definer.

Opening the Metric Definer Web Interface

To start the OVBPI Metric definer Web interface, complete the following steps:

1. Open a new Web browser window.
2. Type the following URL into the Web Browser:

   `http://`*hostname*`:44080/ovbpimetricdefiner`

   where:

   — *`hostname`* is the fully qualified domain name of the system where the OVBPI Server is installed and running. You can use `localhost` as the hostname, if you are starting the definer on the system where the OVBPI server components are installed and running.

   — `44080` is the port number for the Servlet Engine, identified by the `ServletEngine HTTP` port number. Use the port number configured for your system.

   You are prompted for a username and password.

3. Enter the username and password for the Metric definer. Following a new installation, these are:

   — Username: `admin`

   — Password: `ovbpi`

   You can change the password used for the Metric definer; see Appendix B, Servlet Container Authentication.

Closing the Metric Definer

You close the Metric definer by closing the browser Window where the definer is running.

If you close the Metric definer browser Window before completing a definition (clicking the `OK` button on the appropriate pane), the definition is lost. Before closing the Browser Window, make sure that any definition that you are creating appears in the list of definitions within the Metric definer. If you are modifying a definition, allow the definer to respond (usually by taking you to the page where the revised definition is listed), before closing the Browser Window.

You are advised to use the Web Server authentication mechanisms to lock the Web Browser screen after a certain length of time; this increases the level of security for the Metric definer pages. Refer to Appendix B, Servlet Container Authentication for details of the default authentication.

Accessing the Repository Explorer

Before trying to start the OVBPI Repository Explorer, make sure the `Servlet Engine` and the Model Repository components are started. If it is not, you cannot flow data using the Repository Explorer.

Opening the Repository Explorer Web Interface

To start the OVBPI Repository Explorer Web interface, complete the following steps:

1. Open a new Web browser window.
2. Type the following URL into the Web Browser:

   `http://`*hostname*`:44080/ovbpirepositoryexplorer`

   where:

   — *`hostname`* is the fully qualified domain name of the system where the OVBPI Server is installed and running. You can use `localhost` as the hostname, if you are starting the definer on the system where the OVBPI server components are installed and running.

   — `44080` is the port number for the Servlet Engine, identified by the `ServletEngine HTTP` port number. Use the port number configured for your system.

   You are prompted for a username and password.

3. Enter the username and password for the Repository Explorer. Following a new installation, these are:

   — Username: `admin`

   — Password: `ovbpi`

   You can change the password used for the Repository Explorer; see section Appendix B, Servlet Container Authentication.

Closing the Repository Explorer

You close the Repository Explorer by closing the browser Window where the explorer is running. Closing the browser window has no effect on how the Repository Server operates, but if you have not completed an administration task, the task is lost when the browser window is closed.

You are advised to use the Web Server authentication mechanisms to lock the Web Browser screen after a certain length of time; this increases the level of security for the Repository Explorer pages. Refer to Appendix B, Servlet Container Authentication for details of the default authentication.

Accessing the OVBPI Notification Server Administration Console

Before trying to start the OVBPI Notification Server Administration Console, make sure the `Servlet Engine` is started. If it is not, you cannot access the console.

Opening the Web Administration Console

To start the OVBPI Notification Server Web Administration complete the following steps:

1. Open a new Web browser window.
2. Type the following URL into the Web Browser:

   `http://`*hostname*`:44080/ovbpinotifyadmin`

   where:

   — *`hostname`* is the fully qualified domain name of the system where the OVBPI Server is installed and running. You can use `localhost` as the hostname, if you are starting the definer on the system where the OVBPI server components are installed and running.

   — `44080` is the port number for the Servlet Engine, identified by the `ServletEngine HTTP` port number. Use the port number configured for your system.

   You are prompted for a username and password.
3. Enter the username and password for the Web Administration Console, following a new installation, these are:

   — Username: `admin`

   — Password: `ovbpi`

   You can change the password used for the Notification Server Administration console; see Appendix B, Servlet Container Authentication.

Closing the Web Administration Console

You close the Notification Server Administration Console by closing the browser Window where the Console is running. Closing the browser window has no effect on how the Notification Server operates, but if you have not completed an administration task, the task is lost when the browser window is closed.

You are advised to use the Web Server authentication mechanisms to lock the Web Browser screen after a certain length of time; this increases the level of security for the Notification Server Administration pages. Refer to Appendix B, Servlet Container Authentication for details of the default authentication that is provided for the Notification Server.

Accessing the Business Process Dashboard

Before trying to start the OVBPI Business Process Dashboard, make sure that the `Servlet Engine` component is started from the OVBPI Administration Console. If it is not, you cannot access the OVBPI Dashboard.

Opening the Business Process Dashboard

To access the OVBPI Dashboard, complete the following steps:

1. Open a new Web browser window.
2. Type the following URL into the Web Browser:

   `http://`*`hostname`*`:44080/ovbpidashboard2-0`

   where:

   — *`hostname`* is the fully qualified domain name of the system where the OVBPI Server is installed and running. You can use `localhost` as the hostname, if you are starting the definer on the system where the OVBPI server components are installed and running.

   — `44080` is the port number for the Servlet Engine, identified by the `ServletEngine HTTP` port number. Use the port number configured for your system.

   A browser window opens on the `Home` page for the OVBPI Dashboard. You can select to view Flows, Service Impact or Metrics from this Home page.

Closing the Business Process Dashboard

To close the OVBPI Dashboard Web application, close the browser Window where the Dashboard is running.

Accessing the Intervention Client

Before trying to start the OVBPI Intervention Client, make sure that the `Servlet Engine` component is started from the OVBPI Administration Console. If it is not, you cannot access the Intervention Client.

Opening the Intervention Client Web Application

You access the Intervention Client as follows:

1. Open a new Web browser window
2. Type the following URL:

   `http://`*`hostname`*`:44080/ovbpiintclient`

   where:

   — *`hostname`* is the fully qualified domain name of the system where the OVBPI Server is installed and running. You can use `localhost` as the hostname, if you are starting the definer on the system where the OVBPI server components are installed and running.

   — `44080` is the port number for the Servlet Engine, identified by the `ServletEngine HTTP` port number. Use the port number configured for your system.

   You are prompted for a username and password.
3. You are presented with a dialog to enter the login credentials for the Intervention Client. Enter a User Name and Password for the Client. By default the User Name is `admin` and the Password is `ovbpi`.

   Section Security and the Intervention Client on page 256 describes how to change the username and password credentials for the Intervention Client.

Closing the Intervention Client Web Application

To close the Intervention Client Web application, close the browser Window where it is running.

If you want to increase the level of security for the Intervention Client, you can use the Web Server authentication mechanisms to lock the Web Browser screen after a certain length of time. This is advisable as someone could inadvertently use the Intervention Client to delete flows and flow instances within your OVBPI system. Section Security and the Intervention Client on page 256 provides details of the authentication that is provided for the Intervention Client by default.

Log File Locations and Changing the Log Viewer

All the OVBPI components have log files; however the log files presented through the Status pane relate only to the OVBPI Server components, which are managed using the Administration Console, through an Administration Server.

The Modeler is not an OVBPI server component and its log file is described in Chapter 6, OVBPI Modeler Administration. The structure of all OVBPI log files is described in the *OpenView Business Process Insight Problem Solving Guide*.

All the OVBPI log files, including the Modeler log files, are located in the following directory:

`OVBPI-install-dir\data\log`

Use the `View Log` button on the `Status` pane (adjacent to the Start and Stop buttons) to view the log files for the server components. By default the logs are displayed using the following commands:

- `OVBPI-install-dir/lbin/bia/logviewer.sh` (HP-UX)
- `notepad` (Windows)

Section Component Configurations - Logging on page 131 describes the parameters for modifying the viewer application that is used to display the log files for OVBPI Server components.

Component Configurations - Notification Server

Table 1 lists the parameters that you can modify to change the configuration of the Notification Server using the Administration Console.

Table 1 Notification Server Parameters

Descriptive Parameter Name	Description
SMTP hostname	The fully qualified domain name of the system where the SMTP email server is running. The OVBPI system needs access to a running SMTP email server in order to send impact alerts through email to registered users. Refer to Chapter 9, Notification Server Configuration for information about registering users to receive email alerts.
SMTP port number	The port number that the SMTP server is communicating on. The OVBPI system communicates with the SMTP email server through a specific port number, which is usually port 25.
Sender email address	The email address that you want to appear in the `Sender:` field on email notifications sent from the OVBPI system. This is the return email address that is sent out in response to impact alerts for all email messages. This is required for email messages sent out in response to impact alerts.
Retry interval to SMTP server (minutes)	The time (in minutes) that the Notification Server waits before attempting to send a notification to the SMTP server, if it fails on the first attempt. If the Notification Server is unable to send the message, it logs a warning message and temporarily stores the message in the database. The message is removed from the database when the notification is successfully delivered.

Table 1 Notification Server Parameters

Descriptive Parameter Name	Description
Maximum retries to SMTP server	Maximum number of retry attempts made by the Notification Server to send a notification message to the SMTP server. When the maximum number of retries is reached, the Notification Server deletes the message from the database, issues an error message in its log file and makes no further attempts to deliver the message to the email server.
Retry interval to OpenView Operations (minutes)	The time (in minutes) that the Notification Server waits before attempting to send an OpenView message to OpenView Operations, if it fails on the first attempt. If the Notification Server is unable to send the message, it logs a warning message and temporarily stores the message in the database. The warning message is removed from the database when the notification is successfully delivered.
Maximum retries to OpenView Operations	Maximum number of retry attempts made by the Notification Server to send an OpenView message to OpenView Operations. When the maximum number of retries is reached the Notification Server deletes the message from the database, issues an error message in its log file and makes no further attempts to deliver it.

Table 1 Notification Server Parameters

Descriptive Parameter Name	Description
Retry interval for script execution (minutes)	The time (in minutes) that the Notification Server waits before attempting to execute a script that you have created, if the script fails to execute on the first attempt. If the Notification Server is unable to execute a script, it logs a warning message in the Notification Server log file.
Maximum retries for script execution	Maximum number of retry attempts made by the Notification Server to execute a script that you have created. When the maximum number of retries is reached the Notification Server, issues an error message in its log file and makes no further attempts to execute the script.
Timeout for script execution (minutes)	The time (in minutes) that the Notification Server waits before aborting the execution of the script, after the script has started executing. Once aborted, the Notification Server waits for the retry interval before attempting to execute the script again.

Changing the Notification Server Parameters

To change the Notification Server parameters, complete the following steps on the Windows system, where the OVBPI Server is installed:

1. Start the Administration Console as described in section Administration Console Description on page 60.
2. Select the `Component Configuration > Notification Server` option from the Navigator pane on the OVBPI Administration Console.
3. Enter the changes that you want to make to the Notification Server parameters in the right-hand pane as appropriate.
4. Click `Apply` to apply the changes to the OVBPI configuration.

5. Select the `Status` option to move to the panel where the OVBPI component status are shown.

6. Stop and restart all the OVBPI Server components using the `Stop All` and `Start All` buttons. You could stop and restart only the Notification Server and the Servlet Engine; however, you are recommended to use `Stop All` and `Start All` to make sure that components are started and stopped in the correct order.

The new Notification Server parameter values are now applied to your OVBPI system.

If you want to change the login password for the Notification Server Administration Console, refer to Appendix B, Servlet Container Authentication.

Component Configurations - Engine

The Business Impact Engine configuration parameter settings are divided into a number of logical sections. These sections appear hierarchically in the Administration Console under the `Engine` option as follows:

- Engine Java Virtual Machine (JVM) settings
- Engine Model Cleaner settings
- Engine Instance Cleaner settings
- Engine Event settings
- Engine Notification settings
- Engine JDBC settings

The parameters relating to these settings appear on the right-hand pane of the Administration Console when you select one of these options in the console's navigation tree. If the options are not visible, you need to expand the entries under `Engine` using the usual Explorer style navigation techniques.

Engine Java Virtual Machine (JVM) Settings

This section describes the parameters that you can modify through the `Engine Java Virtual Machine (JVM) settings` option.

Table 2 shows the Business Impact Engine parameters that enable you to modify the amount of memory heap available to the JVM for the system.

A heap is a storage management structure for tracking and allocating memory. In this case, the Java heap is used for allocating the Java objects used by the Engine.

Table 2 Engine Java Virtual Machine (JVM) Settings

Descriptive Parameter Name	Description
Initial size of the JVM heap (MB)	The initial size of the storage allocated by the JVM for Java objects.
Maximum size of the JVM heap (MB)	The maximum size of the storage allocated by the JVM for Java objects.

Modifying the Engine Java Virtual Machine (JVM) Settings

To change the `Engine Java Virtual Machine (JVM) Settings`, complete the following steps on the Windows system, where the OVBPI Server is installed:

1. Start the Administration Console as described in section Administration Console Description on page 60.
2. Select the `Component Configuration > Engine > Engine Java Virtual Machine (JVM) settings` option from the Navigator pane on the OVBPI Administration Console.
3. Enter the changes that you want to make to the settings as appropriate in the right-hand pane.
4. Click `Apply` to apply the changes to the property files.
5. Select the `Status` option to move to the panel where the OVBPI component status are shown.
6. Stop and restart all the OVBPI Server components using the `Stop All` and `Start All` options.

The new Business Impact Engine settings are now applied to your OVBPI system.

Engine Model Cleaner Settings

This section describes the parameters that you can modify through the `Engine Model Cleaner Settings` option. The Model Cleaner is a thread that runs at intervals and deletes Flow and Data definitions that have been undeployed or superseded, and that no longer have any associated instances running.

Table 3 shows the Business Impact Engine parameters that enable you to remove undeployed Flow and Data definitions, and unused Service definitions, from the Business Impact Engine database.

Table 3 Engine Model Cleaner Settings

Descriptive Parameter Name	**Description**
Delay between checking for all Models (minutes)	The time interval (in minutes) that the Business Impact Engine waits before reading the latest list of definitions. This list is used as input for the `Delay between checking each model` parameter.
Delay between checking each Model (seconds)	The time (in seconds) that the Business Impact Engine waits between checking each entry in the list of definitions created as a result of the `Delay between checking for all models` parameter. It is used to establish those definitions that have been undeployed and which are therefore candidates to be deleted. Definitions are not deleted if: • the definition is deployed. • instances of the definition exist within the Business Impact Engine database. • the definition is referenced from another definition that is currently defined in the Business Impact Engine database; for example, a Service Definition is referenced from a Flow definition.
Disable cleanup of files for Models not in database?	A check box indicating whether or not to remove the Java source, Java Class and script files for flow and data models that have been cleaned from the Business Impact Engine. When checked, the source, classes and scripts are not removed.

Modifying the Engine Model Cleaner Settings

To change the `Engine Model Cleaner Settings`, complete the following steps on the Windows system, where the OVBPI Server is installed:

1. Start the Administration Console as described in section Administration Console Description on page 60.
2. Select the `Component Configuration > Engine > Engine Model Cleaner settings` option from the Navigator pane on the OVBPI Administration Console.
3. Enter the changes that you want to make to the Engine settings as appropriate in the right-hand pane.
4. Click `Apply` to apply the changes to the property files.
5. Select the `Status` option to move to the panel where the OVBPI component status are shown.
6. Stop and restart all the OVBPI Server components using the `Stop All` and `Start All` options.

The new Business Impact Engine settings are now applied to your OVBPI system.

Engine Flow and Data Instance Cleaner Settings

This section describes the parameters for the Business Impact Engine Instance Cleaner settings. For details of the Metric Engine Instance Cleaner settings, refer to section Metric Engine Instance Cleaner Settings on page 98.

The Business Impact Engine Instance Cleaner settings section controls how often the instance cleaner thread is run and therefore how often the Completed and Active instances are deleted (or not deleted) from the database. Chapter 10, Intervention provides more information on the business impact Engine instance cleaner thread and the parameters that control it.

Instances are initially marked as being candidates to be deleted and the business impact Engine instance cleaner then runs at intervals that you specify to delete the instances that have been marked as candidates. More details of this process are provided in Engine Flow and Data Instance Cleaner Parameters on page 257.

You also have the option to delete complete instances as soon as they are complete using this setting. There is no requirement for the Engine Instance Cleaner to run in this case, as the instances are deleted as soon as they are complete.

For a new installation, the Engine Instance Cleaner thread does not run and instances are not deleted when they complete. Completed and Active instances can therefore accumulate in the database and this might have an impact on the performance of your OVBPI system.

If performance is a consideration, you need to modify this default to remove these instances and also consider archiving data in order that you can monitor historical data. Refer to Chapter 10, Intervention for details of how you can specify your own SQL to archive flow data before it is removed from the OVBPI database.

There are settings for how often the Engine Instance Cleaner marks instances as being candidates to be deleted, plus individual settings for controlling when the Completed and Active Instances are deleted. These are described in Table 4, Table 5 and Table 6.

➤ If you delete all Completed instances as soon as they are completed, data is no longer available to be viewed through the Business Process Dashboard for monitoring or archiving purposes.

Table 4 lists the Engine Flow and Data Instance Cleaner thread settings, which enable you to control how often the instance cleaner thread is executed.

Table 4 Engine Flow and Data Instance Cleaner Settings

Descriptive Parameter Name	**Description**
Mark instances for deletion:	The option for marking flow and data instances as candidates to be deleted by the Instance Cleaner thread at a specified interval (`Delete interval`). You have the following options: • `Never` - instances are never marked to be deleted. • `Periodically` - instances are marked to be deleted at the interval that you specify. • `Once a day` - instances are marked to be deleted once a day, at the time that you specify. • `Twice a day` - instances are marked to be deleted twice a day, at the times that you specify.
Delete interval	The time interval used to control when the instance cleaner thread runs to delete any instances that have been marked for deletion.
Delete batch size	The maximum number of active and complete flow and data instances that the instance cleaner thread deletes in one `Delete interval`. There are four batches, one each for active and complete flow instances, and active and complete data instances.

Table 5 lists the settings that control when Completed flow and data instances are deleted from the Business Impact Engine database.

Table 5 Completed Flow and Data Instance Cleaner Settings

Descriptive Parameter Name	**Description**
Delete completed instances from the database	The option for deleting Completed instances from the database. You have the following options for deleting Completed instances: • `Never` - completed instances are never deleted and remain in the database • `As scheduled above` - completed instances are deleted from the database at an interval that you specify in the `Engine Flow and Data Instance Cleaner settings` options. • `Immediately on completion` - completed instances are deleted as soon as they reach the state `COMPLETED` and are not under the control of the settings for the Engine Instance Cleaner thread.
Age of the completed instances to be removed (minutes)	The age (in minutes) that the Completed instances must be before they can be deleted. This is the age of the instance at the time when the Engine Instance Cleaner runs. The following are quick references to common time durations expressed as minutes: 1 day = 1440 minutes 1 week (7 days) = 10080 minutes

Table 6 lists the settings that control how often Active flow and data instances are deleted from the Business Impact Engine database.

Table 6 Active Flow and Data Instances Cleaner Settings

Descriptive Parameter Name	Description
Delete active instances from the database?	This is an indication of whether or not you want to delete active instances from the database. • `Never` - active instances are never deleted and remain in the database • `As scheduled above` - active instances are deleted from the database at an interval that you specify in the `Engine Flow and Data Instance Cleaner settings` options.
Age of the active instances to be removed:	The age (in days, hours or minutes) that the Active instances must be before they can be deleted. This is the age of the instance at the time when the Engine Instance Cleaner runs. If you enter a value of 60 minutes, the next time that you access this pane, the value is presented in terms of hours and not minutes.

These parameters, and information about archiving instances before deleting them, are further described in section Engine Flow and Data Instance Cleaner Parameters on page 257.

Modifying the Engine Instance Cleaner Settings

To change the Engine Instance Cleaner Settings, complete the following steps on the Windows system where the OVBPI Server is installed:

1. Start the Administration Console as described in section Administration Console Description on page 60.
2. Select the `Component Configuration > Engine > Engine Flow and Data Instance Cleaner settings` option from the Navigator pane on the OVBPI Administration Console.
3. Enter the changes that you want to make to the Engine Instance Cleaner settings as appropriate in the right-hand pane.

4. Click `Apply` to apply the changes to the property files.
5. Select the `Status` option to move to the panel where the OVBPI component status are shown.
6. Stop and restart all the OVBPI Server components using the `Stop All` and `Start All` options.

The new Business Impact Engine settings are now applied to your OVBPI system.

Engine Event Settings

This section describes the parameters that you can modify through the `Engine Event settings` option.

The parameter settings listed in Table 7 enable you to minimize the amount of time that the components spend in loops and possible database deadlock situations and provides settings for throughput rates.

Table 7 Engine Event Settings

Descriptive Parameter Name	Description
Maximum event generation number	A limit for the number of child events created from an initial incoming event. This is intended to interrupt potential infinite loops for subscriptions to data definitions.
Maximum retries for an event transaction deadlock	The maximum number of attempts by the Business Impact Engine to retry a database transaction before aborting it. It is possible for deadlocks to occur when the Engine and other applications are accessing the OVBPI database simultaneously. This parameter ensures that, in the case of a deadlock, the Engine aborts the transaction in order to break the deadlock. When this threshold is exceeded, the Business Impact Engine generates an error message and sends an exception to the Business Event Handler. The Business Event Handler then rolls back the event transaction and retries at a later time.

Table 7 Engine Event Settings

Descriptive Parameter Name	**Description**
Event sample count used for calculating rates[1]	The number of Events used as a basis for calculating throughput rates. Set this parameter according to the incoming Event rate (to the Business Impact Engine). You want to use sufficient numbers of Events to give you a realistic throughput rate.
Decay period for rates when no events are received (seconds)[1]	The time period after which the throughput rates are decremented, in stages, to zero. This is used for periods of inactivity, to maintain realistic results for throughput rates.

1. These parameters are relevant to the mechanism for generating business metrics in versions of OVBPI before version 02.00. In versions of OVBPI prior to version 02.00, business metric data was generated within the Business Impact Engine and not the Metric Engine. The parameters exist for backwards compatibility for use with pre-version 02.00 business metrics. You are advised not to use them in new implementations as they might be removed in a future version of the product.

Modifying the Engine Event Settings

To change the Engine Event Settings, complete the following steps on the Windows system, where the OVBPI Server is installed:

1. Start the Administration Console as described in section Administration Console Description on page 60.
2. Select the `Component Configuration > Engine > Engine Event settings` option from the Navigator pane on the OVBPI Administration Console.
3. Enter the changes that you want to make to the Engine settings as appropriate in the right-hand pane.
4. Click `Apply` to apply the changes to the OVBPI configuration.
5. Select the `Status` option to move to the panel where the OVBPI component status are shown.
6. Stop and restart all the OVBPI Server components using the `Stop All` and `Start All` options.

The new Business Impact Engine settings are now applied to your OVBPI system.

Engine Notification Settings

The following are the parameters that you can modify through the `Engine Notification settings` option.

If the Business Impact Engine initially fails to send an event notification to the Notification Server, it can try again. The settings listed in Table 8 on page 90 control at what impact level notifications are sent, and the number of retry attempts and the interval between the retries for the notifications.

Table 8 Engine Notification Settings

Descriptive Parameter Name	Description
Node impact notification service severity threshold	The minimum threshold for the severity for a node's service that can trigger an impact of impeded or blocked for the flow. When one (or more) node services reach the threshold specified a notification event is issued. Possible levels of severity are: Normal, Warning, Minor, Major and Critical. These map to other OpenView products' levels of severity.
Maximum number of retries	The maximum number of attempts that the Business Impact Engine tries to send a notification event to the Notification Server, if the initial attempt fails. When the maximum number of retries is reached, the Business Impact Engine issues an error message in its log file and makes no further attempts to send the notification event.
Retry delay (seconds)	The delay (in seconds) between the Business Impact Engine's attempts to retry sending notification events to the Notification Server. Each time the Engine fails a retry attempt, it issues a warning message to the log file.

Modifying the Engine Notification Settings

To change the Engine Notification Settings, complete the following steps on the Windows system, where the OVBPI Server is installed:

1. Start the Administration Console as described in section Administration Console Description on page 60.
2. Select the `Component Configuration > Engine > Engine Notification settings` option from the Navigator pane on the OVBPI Administration Console.
3. Enter the changes that you want to make to the Business Impact Engine settings as appropriate in the right-hand pane.
4. Click `Apply` to apply the changes to the OVBPI configuration.
5. Select the `Status` option to move to the panel where the OVBPI component status are shown.
6. Stop and restart all the OVBPI Server components using the `Stop All` and `Start All` options.

The new Business Impact Engine settings are now applied to your OVBPI system.

Engine JDBC Settings

The following are the parameters that you can modify through the `Engine JDBC settings` option.

Table 9 lists the settings that enable you to tailor the JDBC connection between the Business Impact Engine and the database.

Table 9 Engine JDBC Settings

Descriptive Parameter Name	Description
Maximum number of active JDBC Connections	The maximum number of JDBC connections that the Business Impact Engine can have active at any one time. If a connection does not become available in the required time, the connection is refused and the Engine is unable to commit the transaction. In this case, the Engine writes an error to its log file.
Maximum wait time for a JDBC Connection (seconds)	The maximum length of time (in seconds) that the Business Impact Engine waits for an available JDBC connection before reporting an error. If a connection does not become available in the required time, the connection is refused and the Engine is unable to commit the transaction. In this case, the Engine writes an error to its log file.
Maximum number of idle JDBC Connections[1]	The maximum number of JDBC connections that can be idle at any one time.
Maximum number of active JDBC Prepared Statements[1]	The maximum number of database Prepared Statements that can be active at any one time.
Maximum wait time for a JDBC Prepared Statement (seconds)	The maximum length of time (in seconds) that the Business Impact Engine waits for a Prepared Statement to be executed before reporting an error in the log file and aborting the transaction.
Maximum number of idle JDBC Prepared Statements[1]	The maximum number of Prepared Statements that can be idle at any one time.

1. These connections are managed by Apache Commons DBCP (Database Connection Pool).

Modifying the Engine JDBC Settings

To change the Engine JDBC Settings, complete the following steps on the Windows system, where the OVBPI Server is installed:

1. Start the Administration Console as described in section Administration Console Description on page 60.
2. Select the `Component Configuration > Engine > Engine JDBC settings` option from the Navigator pane on the OVBPI Administration Console.
3. Enter the changes that you want to make to the Engine settings as appropriate in the right-hand pane.
4. Click `Apply` to apply the changes to the OVBPI configuration.
5. Select the `Status` option to move to the panel where the OVBPI component status are shown.
6. Stop and restart all the OVBPI Server components using the `Stop All` and `Start All` options.

The new Business Impact Engine settings are now applied to your OVBPI system.

Component Configurations - Metric Engine

The Metric Engine configuration parameter settings are divided into a number of logical sections. These sections appear hierarchically in the Administration Console, under the `Metric Engine` option as follows:

- Metric Engine Event settings
- Metric Engine Threshold settings
- Metric Engine Instance Cleaner settings

The parameters relating to these settings appear on the right-hand pane of the Administration Console when you select one of the Metric Engine options in the console's navigation tree. If the Metric Engine options are not visible in the navigation tree, expand the entries under Metric Engine using the usual Explorer-style navigation techniques.

Metric Engine Event Settings

This section describes the parameters that you can modify through the Metric Engine Event settings option. These parameters enable you to control how often the Metric Engine polls the OVBPI database in order to obtain the metric and statistics data to write to the Metric database tables. The parameters also control how far back in time the metric statistics are calculated, following a system shutdown.

When OVBPI is restarted following a system shutdown, the metric statistics are calculated for all the metric definitions defined for flows. If you have shut your OVBPI system down for a significant amount of time (several days), then these statistic calculations can take considerable time, and impact the performance of your system. You can configure how far back in time to calculate these statistics following a restart using the Metric Engine Event settings options.

The Metric Engine Event Settings are divided into two logical sections. These sections appear on the `Metric Engine Event settings` pane within the following categories:

- Metric Engine Event Settings
- Metrics Statistics Backfill settings

Table 10 lists the parameters that you can set to configure how often the Metric Engine polls the OVBPI database.

Table 10 Metric Engine Event Settings

Descriptive Parameter Name	Description
Metric event polling interval (seconds)	The time interval (in seconds) that the Metric event processor polls for metric events, which you have defined in the Metric definer, from the OVBPI database in order to add them to the Metric database tables.
Statistics generation polling interval (seconds)	The time interval (in seconds) that the Statistics generator polls the metric tables in order to update the statistics that have been configured using the Metric Definer.

Table 11 lists the parameters that you can set to configure how far back in time the Metric Engine is to calculate Metric statistics following the restart of the OVBPI server components.

Table 11 Metric Engine Backfill Settings

Descriptive Parameter Name	Description
Maximum age of generated statistics on startup (days)	The time interval (in days) that the Metric Engine uses to calculate historical Metric statistics following a period of OVBPI shutdown.

Modifying the Metric Engine Event Settings

To change the Metric Engine Event Settings, complete the following steps on the Windows system where the OVBPI Server is installed:

1. Start the Administration Console as described in section Administration Console Description on page 60.
2. Select the `Component Configuration > Metric Engine > Metric Engine Event settings` option from the Navigator pane on the OVBPI Administration Console.
3. Enter the changes that you want to make to the Metric Engine Event settings as appropriate in the right-hand pane.

4. Click `Apply` to apply the changes to the property files.
5. Select the `Status` option to move to the panel where the OVBPI component status are shown.
6. Stop and restart the Metric Engine component.

The new Metric Engine settings are now applied to your OVBPI system.

Metric Engine Threshold Settings

This section describes the parameters that you can modify through the Metric Engine Threshold settings option.

Table 12 lists the Metric Engine Threshold settings that enable you to control how often the Metric Engine polls its data in order to identify threshold information for alarms, notifications and for statistical sampling.

Table 12 Metric Engine Threshold Settings

Descriptive Parameter Name	**Description**
Threshold polling interval (seconds)	The time interval (in seconds) that the Threshold checker polls for metric values from the Metric database in order to determine whether an alarm or a notification needs to be generated.
Threshold minimum sample count, used for calculating relative thresholds	The minimum number of metric instances required for the calculation of the relative scope thresholds. If the number of metric instances available for sampling is less than this minimum figure, the threshold is not calculated. As a result, there are no notifications or alarms generated based on the threshold.

Modifying the Metric Engine Threshold Settings

To change the Metric Engine Threshold Settings, complete the following steps on the Windows system where the OVBPI Server is installed:

1. Start the Administration Console as described in section Administration Console Description on page 60.
2. Select the `Component Configuration > Metric Engine > Metric Engine Threshold settings` option from the Navigator pane on the OVBPI Administration Console.
3. Enter the changes that you want to make to the Metric Engine Threshold settings as appropriate in the right-hand pane.
4. Click `Apply` to apply the changes to the property files.
5. Select the `Status` option to move to the panel where the OVBPI component status are shown.
6. Stop and restart the Metric Engine component.

The new Metric Engine settings are now applied to your OVBPI system.

Metric Engine Instance Cleaner Settings

This section describes the parameters that you can modify through the Metric Engine Instance Cleaner settings option.

The Metric Engine instance cleaner settings section controls how often the Metric Engine instance thread is run and therefore how often Active and Completed metric instances, Metric alarm instances and statistics are deleted.

If you set a low Collection interval for your business process metric definitions, the metric data can accumulate very quickly and consume a considerable amount of disk space. In addition, if you shutdown your OVBPI system and restart it after a significant amount of time, the metric statistics are calculated for the period of the shutdown as soon as the OVBPI Server is restarted. As an example, if you set a Collection interval of five minutes and shut down your OVBPI system for a week. When you restart your OVBPI Server, the Metric Engine calculates all the metric statistics for each five-minute period since the last time the metrics were calculated. This can result in a significant amount of calculation time, and a significant amount of disk space for the results.

If these results are immediately deleted as a result of the instance cleaner settings, this can also have an impact on the overall operational performance of your machine.

The Collection interval is a parameter that you set when you define a business process metric in the Metric definer and is described in the *OpenView Business Process Insight Concepts and Modeling Flows*.

The configuration parameters are divided into a number of logical sections. These sections appear in the Metric Engine Instance Cleaner settings pane within the following categories:

- Metric Engine Instance Cleaner settings
- Active Metric Instances settings
- Completed Metric Instances settings
- Metric Statistics settings
- Metric Alarm Instances settings

The parameters relating to these settings are described in the following sections.

Metric Engine Instance Cleaner Settings

Table 13 lists the settings that controls how often the Metric Engine instance cleaner thread is executed. This interval impacts all the other sections on the Metric Instance Cleaner pane.

Table 13 Metric Engine Instance Cleaner Settings

Descriptive Parameter Name	Description
Instance cleaner execution interval (minutes)	The time period (in minutes) that you want the Metric instance cleaner thread to run. When it runs, it deletes all eligible Active metric instances, Completed metric instances, Metric Alarm instances and Statistics.

Active Metric Instances Settings

Table 14 lists the settings that enable you to control the amount of Active metrics instance data held in the Metrics database.

Table 14 Active Metric Instances Settings

Descriptive Parameter Name	Description
Delete active metric instances from the database?	Select this check box if you want to delete Active metric instances from the Metrics database. Clear the check box if you do not want to delete the Active metric instances.
Age of active metric instances to be removed (days)	This option is available when you select the check box for deleting Active metric instances. Enter the age (in days) of the Active metric instances that you want to be deleted when the Metric Engine instance cleaner thread is run.

Completed Metric Instances Settings

Table 15 lists the settings that enables you to control the amount of Completed metrics instance data held in the Metrics database.

Table 15 Completed Metric Instances Settings

Descriptive Parameter Name	Description
Delete completed metric instances from the database?	Select this check box if you want to delete Completed metric instances from the Metrics database. Clear the check box if you do not want to delete the Completed metric instances.
Age of completed metric instances to be removed (days)	This option is available when you select the check box for deleting Completed metric instances. Enter the age (in days) of the Completed metric instances that you want to be deleted when the Metric Engine instance cleaner thread is run

Metric Statistics Settings

Table 16 lists the settings that enable you to control the amount of statistical data held in the Metrics database.

Table 16 Metric Statistics Settings

Descriptive Parameter Name	Description
Delete metric statistics from the database?	Select this check box if you want to delete statistics from the Metrics database. Clear the check box if you do not want to delete the statistics data.
Age of metric statistics to be removed (days)	This option is available when you select the check box for deleting statistics. Enter the age (in days) of the statistics data that you want to be deleted when the Metric Engine instance cleaner thread is run

Metric Alarm Instances Settings

Table 17 lists the settings that enables you to control the amount of Metric Alarm instance data held in the Metrics database.

Table 17 Metric Alarm Instances Settings

Descriptive Parameter Name	Description
Delete metric alarm instances from the database?	Select this check box if you want to delete Metric Alarm instances from the Metrics database. Clear the check box if you do not want to delete the Metric Alarm instances.
Age of metric alarm instances to be removed (days)	This option is available when you select the check box for deleting Completed metric instances. Enter the age (in days) of the Metric Alarm instances that you want to be deleted when the Metric Engine instance cleaner thread is run

Modifying the Metric Engine Instance Cleaner Settings

To change the Metric Engine Instance Cleaner Settings, complete the following steps on the Windows system where the OVBPI Server is installed:

1. Start the Administration Console as described in section Administration Console Description on page 60.
2. Select the `Component Configuration > Metric Engine > Metric Engine Instance Cleaner settings` option from the Navigator pane on the OVBPI Administration Console.
3. Enter the changes that you want to make to the Metric Engine Instance Cleaner settings as appropriate in the right-hand pane.
4. Click `Apply` to apply the changes to the property files.
5. Select the `Status` option to move to the panel where the OVBPI component status are shown.
6. Stop and restart the Metric Engine component.

The new Metric Engine settings are now applied to your OVBPI system.

Component Configurations - OVIS Interoperability

The `OVIS interoperability` options enable you to configure interoperability with OpenView Operations Internet Services. The configuration options described in this section do not apply to the configuration of the OVBPI custom probes for OVIS; they apply only to OVBPI polling OVIS for details of operational services and impact events.

If you choose to import services, you must also modify other characteristics of the OVIS connection to enable OVBPI to communicate with OVIS.

The OVIS interoperability configuration parameter settings are divided into a number of logical sections. These sections appear on the `OVIS interoperability` pane within the following categories:

- Enable/Disable
- Event poll intervals
- OVIS database settings
- Service Import settings

The parameters relating to these categories are described in the following sections.

Enable/Disable

Table 18 lists the settings that enable you to configure whether or not you want OVBPI to integrate with OVIS.

Table 18 Enable/Disable

Descriptive Parameter Name	Description
Enable OVIS interoperability?	Check this check box to if you want OVBPI to integrate with OVIS. If the check box is not selected, operational services can not be imported and OVBPI cannot generate service impact events. In addition, you are not able to modify any other configuration options for the `OVIS interoperability` Component Configuration.

Event Poll Intervals

Table 19 lists the setting that enable you to configure the time interval that OVBPI uses to for polling OVIS for alarms, service impact events and violations.

Table 19 OVIS Event Poll Intervals

Descriptive Parameter Name	Description
Alarm polling interval (minutes)	The time interval (in minutes) for the poll interval where the OVIS Event Receiver polls for alarm events from OpenView Internet Services.
SLO violation polling interval (minutes)	The time interval (in minutes) for the poll interval where the OVIS Event Receiver polls for SLO violation events from OpenView Internet Services.
SLA violation polling interval (minutes)	The time interval (in minutes) for the poll interval where the OVIS Event Receiver polls for SLA violation events from OpenView Internet Services.

OVIS Database Settings

These settings enable you to specify the details for the OVIS database in order that OVBPI can access alerts, SLA and SLO violations; the Business Impact Engine polls the OVIS database and the Business Process Dashboard queries the OVIS database.

You need to find out these details from the person responsible for managing OVIS and its database.

Table 20 lists the parameters that enable you to specify the OVIS database details.

Table 20 OVIS Database Settings

Descriptive Parameter Name	Description
Database type	Either MS SQL Server or Oracle.
Database server hostname	The fully qualified host name of the system where the Reporter database used by OVIS is installed.

Table 20 OVIS Database Settings

Descriptive Parameter Name	Description
Database server port number	The port number for the database used by OVIS.
Database user name	The username that OVBPI needs to specify to access the OVIS tables in the database.
Password	The password that OVBPI needs to specify to access the OVIS tables in the database.
Database name[1] (MS SQL only)	The name of the database for the OVIS data within the Microsoft SQL Server database.
Database SID[2] (Oracle only)	The database instance for the OVIS data within the Oracle database.

1. Available only when OVIS configured for a Microsoft SQL Server database
2. Available only when OVIS configured for an Oracle Server database

Service Import Settings

This section describes the parameters that you can modify through the `Service Import settings` option.

The `Service Import settings` must be applicable to the OVIS Server that uses the database described in section OVIS Database Settings on page 103. You cannot specify Service Import settings for one OVIS Server and have `OVIS Database settings` that are applicable to another OVIS Server.

Table 21 lists the setting that enable you to configure the location of the OVIS system from where you want to import service definitions when you are designing business flows using the OVBPI Modeler.

Table 21 OVIS Service Import Settings

Descriptive Parameter Name	Description
OVIS management server hostname	The fully qualified host name of the system where the OpenView Internet Services Management Server is installed.
OVIS HTTP server port number	The port number for the Internet Information Services (IIS) Web Server used by OVIS.

Modifying the OVIS Interoperability Settings

To change the `OVIS interoperability` settings, complete the following steps on the Windows system, where the OVBPI Server is installed:

1. Start the Administration Console as described in section Administration Console Description on page 60.
2. Select the `Component Configuration > OVIS interoperability option` from the Navigator pane on the OVBPI Administration Console.
3. Enter the changes that you want to make to the settings as appropriate in the right-hand pane.
4. Click `Apply` to apply the changes to the OVBPI configuration.
5. Select the `Status` option to move to the panel where the OVBPI component status are shown.
6. Stop and restart all the OVBPI Server components using the `Stop All` and `Start All` options.

The new settings are now applied to your OVBPI system.

Component Configurations - OVO Interoperability

The OpenView Operations option enables you to specify whether or not you want OVBPI to interoperate with OpenView Operations Service Navigator (on HP-UX), or OpenView Operations for Windows.

➤ An OVBPI Server can connect to one OpenView Operations Server at any one time, this can be on HP-UX (OpenView Service Navigator), or Windows (OpenView Operations for Windows); it cannot connect to both.

Table 22 lists the OVO parameters.

Table 22 OpenView Operations

Descriptive Parameter Name	Description
Enable OpenView Operations interoperability?	Select this check box to enable OVBPI to import operational services from OpenView Operations and to receive impact events. If the check box is not selected, operational services are not synchronized with those defined in the Modeler, and you are not able to modify any other configuration options for the `OVO interoperability` Component Configuration, or deploy a Flow definition that references an OVO Service definition.
OpenView Operations Adaptor hostname	The fully qualified domain name for the system where the OVO Adaptor is installed and running.
Clients' maximum wait time (seconds)	The maximum time (in seconds) that a client waits for a response from the adaptor before assuming that an error has occurred and causing the connection to time out. An error message is written to its log file when this maximum time is exceeded.
Clients' reconnect interval (seconds)	The time (in seconds) that a client waits before retrying a connection to the adaptor following a previously failed connection. A warning message is written to its log file each time the client fails to reconnect.

If you want to enable or disable OVO synchronization; see section Enabling or Disabling OVO Interoperability on page 107.

If you want to modify the Adaptor hostname, refer to section Modifying OVO Adaptor Hostname on page 108.

If you want to modify the client maximum wait time and reconnect intervals, refer to section Modifying the Client Maximum Wait and Reconnect Intervals on page 109.

Enabling or Disabling OVO Interoperability

This setting enables you to configure whether or not you want to enable or disable synchronization with the OVO Adaptor services.

To change modify this setting, complete the following steps:

1. Start the Administration Console as described in section Administration Console Description on page 60.
2. Select the `Component Configuration > OVO interoperability` option from the Navigator pane on the OVBPI Administration Console.
3. Check or uncheck the checkbox labeled Enable OpenView Operations interoperability as appropriate.
4. Click `Apply` to apply the changes to the OVBPI configuration.
5. Select the `Status` option to move to the panel where the OVBPI component status are shown.
6. Stop and restart all the OVBPI Server components using the `Stop All` and `Start All` options.

The new setting is now applied to your OVBPI system.

Modifying OVO Adaptor Hostname

This is the hostname of the system where the OVBPI OVO Adaptor is installed. As the adaptor has to be installed on the same system as the OVO Server, this is also the hostname of the system where either OpenView Service Navigator or OpenView Operations for Windows are installed (according to your configuration). If you want to change this hostname for any reason, complete the following steps on the Windows system, where the OVBPI Server is installed:

1. Make sure that the OVBPI OpenView Operations Adaptor is installed and running on the new system.
2. Make sure that the OVO services required by your OVBPI flows are available on the new server.
3. Use the OVBPI Modeler to undeploy all the parent flows that link to the affected OVO services.

   You do this by individually selecting the Flow Definitions from the navigator pane in the OVBPI Modeler and selecting `Undeploy` from the `File` menu. Chapter 6, OVBPI Modeler Administration discusses undeployment in more detail.
4. Start the Administration Console on the OVBPI Server system as described in section Administration Console Description on page 60.
5. Using the Administration Console, stop the Business Impact Engine and the Model Repository using the `Stop` button on the `Status` option.
6. Select the `OVO interoperability` option from the navigator pane on the OVBPI Administration Console.

   On the right-hand pane, enter the new name for the `OpenView Operations Adaptor hostname`. This should be the fully qualified hostname for the system where either Service Navigator is installed, or OpenView Operations for Windows is installed.
7. Select `Apply` and the changes are made to the configuration files.
8. Select the `Status` option where the component states are listed.

9. Stop and all the OVBPI Server components using the `Stop All` button.

   The OVBPI Modeler issues a warning stating that it has lost its communication link with the OVBPI Server and prompting you to close the modeler. All unsaved changes are saved when the OVBPI Modeler shuts down.

10. Restart the all the OVBPI Server components using the `Start All` button.
11. Restart the OVBPI Modeler so it re-establishes its connection to the Model Repository.
12. Use the `File|Link to OVO Services...` menu option in the Modeler to re link the OVO services on the new OVO Server.
13. Use the OVBPI Modeler to redeploy the previously undeployed Flow and Service definitions to the new OpenView Operations system.

    This is in order that the new services are registered with the OVO system.

You have now completed the re configuration for the hostname for the OpenView Operations Adaptor system.

Modifying the Client Maximum Wait and Reconnect Intervals

To change these settings, complete the following steps on the Windows system, where the OVBPI Server is installed:

1. Start the Administration Console as described in section Administration Console Description on page 60.
2. Select the `OVO interoperability` option from the Navigator pane on the OVBPI Administration Console.
3. Enter the changes that you want to make to the settings as appropriate in the right-hand pane.
4. Click `Apply` to apply the changes to the OVBPI configuration.
5. Select the `Status` option to move to the panel where the OVBPI component status are shown.
6. Stop and restart all the OVBPI Server components using the `Stop All` and `Start All` options.

The new settings are now applied to your OVBPI system.

Component Configurations - Model Repository

The Model Repository configuration has one section as follows:

- Background service import

The parameters relating to these settings appear on the right-hand pane of the Administration Console when you select one of these options in the console's navigation tree.

Background Service Import

This section describes the parameters that you can modify through the `Background service import` option.

These settings enable you to configure the intervals for polling OVIS and OpenView Operations as a background activity.

Services are imported into the Model Repository as a background activity at the time period specified and each time the Model Repository is started. The services are imported only when you have also enabled service definition import for OVIS as described in section Component Configurations - OVIS Interoperability on page 102. OVO Services are references to the Services and are not imported into the Model Repository.

This then makes the services available automatically to you through the Modeler without needing to refresh the services manually through the Modeler interface.

Table 23 lists the Background Service Import parameters that you can modify.

Table 23 Background Service Import

Descriptive Parameter Name	Description
Enable background service definition import?	A check box to enable or disable importing service definitions into the Model Repository. When checked, background import is enabled. If you uncheck this check box, you can use the menu options within the OVBPI Modeler to import the services that you need defined.
Service import period (minutes)	The time interval (in minutes) for importing Service definitions when background importing is enabled.

Modifying the Background Service Import Parameters

To change the `Background service import` option, complete the following steps on the Windows system, where the OVBPI Server is installed:

1. Start the Administration Console as described in section Administration Console Description on page 60.
2. Select the `Component Configuration > Model Repository> Background service import` option from the Navigator pane on the OVBPI Administration Console.
3. Enter the changes that you want to make to the settings as appropriate in the right-hand pane.
4. Click `Apply` to apply the changes to the OVBPI configuration.

5. Select the `Status` option to move to the panel where the OVBPI component status are shown.
6. Stop and restart all the OVBPI Server components using the `Stop All` and `Start All` options. You can stop and restart the Model Repository component; however, you are advised to use `Stop All` and `Start All` to ensure that the OVBPI components are stopped and started in the correct order.

The new settings are now applied to your OVBPI system.

Component Configurations - Business Event Handler

The `Business Event Handler` options enable you to configure the parameters for the Adaptor that sends and receives events between the Business Impact Engine and the Business Event Handler.

These parameters are listed in Table 24.

Table 24 Business Event Handler

Descriptive Parameter Name	**Description**
Log package information?	A check box that when checked means that the Business Event Handler logs package information.
Log thread information?	A check box that when checked means that the Business Event Handler logs thread information.
Log time information?	A check box that when checked means that the Business Event Handler logs time information.
Maximum number of socket source threads	Maximum number of threads that can be created for the adaptor socket source. When this threshold is exceeded, the request for a thread is refused and the Business Event Handler reports a `Connection Refused` error in its log file.
Maximum number of retries to deliver events into Engine	The maximum number of times the Business Event Handler attempts to deliver an event to the Business Impact Engine (using RMI) before rolling back the transaction for the business event.

Table 24 Business Event Handler

Descriptive Parameter Name	Description
Engine event retry delay (seconds)	Time (in seconds) between each attempt to send an event from the Business Event Handler to the Business Impact Engine using RMI.
Maximum number of retries to deliver events from Hospital	The maximum number of times the Business Event Handler attempts to deliver an event from the Event Hospital to the Business Impact Engine. The Business Impact Engine marks particular categories of event errors to result in the event being sent to the event Hospital and marked to be automatically discharged; for example, events that are received out of sequence and where there is no flow or data instance created for the event. If the Business Event Handler does not succeed in delivering the event to the Engine in within the specified number of retries, the event remains in the event Hospital.
Hospital event poll interval (seconds)	Time (in seconds) that the Business Event Handler polls the Event Hospital looking for events that have been marked as ready for discharge and that can be delivered to the Business Impact Engine.

Modifying the Business Event Handler Parameters

To change these settings, complete the following steps on the Windows system, where the OVBPI Server is installed:

1. Start the Administration Console as described in section Administration Console Description on page 60.
2. Select the `Business Event Handler` option from the Navigator pane on the OVBPI Administration Console.
3. Enter the changes that you want to make to the settings as appropriate in the right-hand pane.
4. Click `Apply` to apply the changes to the OVBPI configuration.

5. Select the `Status` option to move to the panel where the OVBPI component status are shown.
6. Stop and restart all the OVBPI Server components using the `Stop All` and `Start All` options. You can stop and restart the `Business Event Handler` component; however, you are advised to use `Stop All` and `Start All` to ensure that the OVBPI components are stopped and started in the correct order.

The new settings are now applied to your OVBPI system.

Component Configurations - Business Process Dashboard

The `Business Process Dashboard` option enables you to manage the parameters of the Business Process Dashboard.

The configuration parameters are divided into a number of logical sections. These sections appear hierarchically in the Administration Console in the `Business Process Dashboard settings` pane within the following categories:

- General settings
- Link to OVSD service calls and incidents

The parameters relating to these settings appear on the right-hand pane of the Administration Console when you select one of the `Business Process Dashboard` options in the console's navigation tree. If the Business Process Dashboard options are not visible in the navigation tree, expand the entries using the Explorer-style navigation techniques.

General Settings

This option enables you to configure parameters related to the behavior of the OVBPI Dashboard. These parameters are listed in Table 25.

Table 25 Business Process Dashboard General Settings

Descriptive Parameter Name	Description
Page refresh delay (seconds)	The time interval (in seconds) that the Business Process Dashboard page uses to set the refresh interval for the Web browser.
Maximum number of retries on database deadlock	The maximum number of attempts by the Business Process Dashboard to retry a database transaction before aborting it. It is possible for deadlocks to occur when the Business Process Dashboard and other applications are accessing the OVBPI database simultaneously. This parameter ensures that, in the case of a deadlock, the Business Process Dashboard aborts the transaction in order to break the deadlock and generates an error message. which is displayed on a Web page. If the Business Process Dashboard aborts the transaction, use the `Refresh` button on your browser to reload the page. This is particularly significant if you are developing your own customized dashboard, and you are using Microsoft SQL Server.
Database deadlock retry delay (seconds)	The time (in seconds) between each attempt to retry a database transaction.
Show service state for a completed node?	A check box indicating whether or not to show the status of services for nodes that have already been completed in the business flow.
Show superseded flows?	A check box indicating whether or not to show superseded flows through the Dashboard.

Modifying the General Settings

To change these settings, complete the following steps on the Windows system, where the OVBPI Server is installed:

1. Start the Administration Console as described in section Administration Console Description on page 60.
2. Select the `General Settings` option from the Navigator pane on the OVBPI Administration Console.
3. Enter the changes that you want to make to the settings as appropriate in the right-hand pane.
4. Click `Apply` to apply the changes to the OVBPI configuration.
5. If you have modified the `Show service state for a completed node?` parameter, you need to stop and restart the Servlet Engine as described in step 6. If you have not modified this parameter, continue at step 8.
6. Select the `Status` option to move to the panel where the OVBPI component status are shown.
7. Stop and restart the Servlet Engine component.
8. If you have modified the `Page refresh delay (seconds)` parameter, and the parameter was previously set to zero (0), you need to refresh the Web browser window where the Business Process Dashboard is running.
9. The new settings are now applied to your OVBPI system.

Link to OVSD Service Calls and Incidents

This option enables you to link the OVBPI Server to an OpenView Service Desk implementation using the OVBPI Dashboard. If this integration is correctly configured, the Business Process Dashboard can link OVBPI impact reports to the related Service Desk service calls and incidents.

The OVSD integration configuration parameter settings are divided into two logical sections. These sections appear on the `Link to OVSD service calls and incidents` pane within the following categories:

- Link to OVSD service calls and incidents
- OVSD Custom Field configuration

Table 26 and Table 27 on page 120 list the OVSD parameters.

Table 26 Link to OVSD Service Calls and Incidents

Descriptive Parameter Name	**Description**
Enable link?	Select this check box to enable the OVBPI Business Process Dashboard to access OVSD incident reports that relate to OVBPI impact reports. When you disable the Service Desk integration, the next JSP page selected from the Business Process Dashboard does not include any links to Service Desk information.
OpenView Service Desk application server hostname	The fully qualified domain name of the OVSD application server on the system where the OVSD implementation that you want to access is running.
OpenView Service Desk user name	The name of the user account set up within the OVSD implementation for OVBPI. This is the user name that OVBPI uses to access OVSD. You are recommended to create an OVSD user account specifically for OVBPI and this user account should: • be a concurrent user • have the role `Helpdesk` Note that the user name that you specify for OVBPI is the OVSD login name for the user and not the display name.
OpenView Service Desk password	The password associated with the user account.

You can now optionally configure the type of integration that you want to configure with OVSD. Chapter 5, OVBPI and OVSD provides details of the different configuration options and how they relate to the parameters listed in Table 27 on page 120.

If you plan to enable custom field support, you can optionally select to refresh the custom field information presented through the Administration Console to make sure it is up to date. Check the `Refresh Custom Fields Names` button on the OVSD interoperability pane to do this.

Table 27 OVSD Custom Field Configuration

Descriptive Parameter Name	**Description**
OpenView Service Desk integration type	An indication of whether you want to configure OVBPI to use the already defined OVO and OVIS service definitions within OVSD (`Automatic`), or whether you want to enable the use of OVSD custom fields (`Custom only`). There are three options: • `Automatic` • `Custom only` • `Both`
Select OpenView Service Desk custom field to use	The custom field that you want to be used within OVSD to hold OVBPI Service names. The list of fields presented includes those already assigned, and those that are currently available (inactive). If there are no fields currently available, you need to find out whether any of the assigned fields can be reused for OVBPI. This option available only when you choose to enable `Custom only` or `Both` as an integration type.

Modifying the Link to OVSD Service Calls and Incidents Settings

To change these settings, complete the following steps on the Windows system, where the OVBPI Server is installed:

1. Start the Administration Console as described in section Administration Console Description on page 60.
2. Select the `Link to OVSD Service Calls and Incidents` option from the Navigator pane on the OVBPI Administration Console.

3. Enter the changes that you want to make to the settings as appropriate in the right-hand pane.

   If you are changing the user name for the OVBPI Service Desk user, make sure the user has the correct profile characteristics as described in the *OVBPI Installation Guide*.

4. Click `Apply` to apply the changes to the OVBPI configuration.
5. Select the `Status` option to move to the panel where the OVBPI component status are shown.
6. Stop and restart the Servlet Engine component.
7. The new settings are now applied to your OVBPI system.

Component Configurations - MS SQL Server Access

The MS SQL Server Access parameters are available only when you select MS SQL as the database from within the OVBPI installation procedure. If you select Oracle as your database, the MS SQL Server parameters are not available through the Administration Console.

The `MS SQL Server Access` option enables you to view the current configurations for the SQL Server access and modify the password for the database access. The database is used by the following components:

- Business Impact Engine
- Metric Engine
- Business Event Handler
- Notification Server
- Business Process Dashboard
- Intervention Client
- OVIS, where you have installed the OVBPI OVIS Custom Probes

These parameters are listed in Table 28.

Table 28 MSSQL Access

Descriptive Parameter Name	Description
MS SQL Server hostname	This is a non-modifiable field and shows the fully qualified domain name of the system where the Microsoft SQL Server Database is installed.
MS SQL Server port number	This is a non-modifiable field and shows the port number that the Microsoft SQL Server Database is expecting connections on.
MS SQL Server database	This is a non-modifiable field and shows the name of the database configured for OVBPI.
MS SQL Server login user	This is a non-modifiable field and shows the name of the user who has permission to read and write the OVBPI database files.

Table 28 MSSQL Access

Descriptive Parameter Name	Description
MS SQL Server login password	The password for the user with permission to read and write the OVBPI database files. This field can be used to resynchronize the OVBPI version of the password with the database password after the database password is changed using the database management tools.
Confirm password	Confirmation of the SQL Server login password.

To change the password for the database, follow the instructions on section Changing the Password Details on page 123.

Changing the Password Details

This section describes how to change the password for the OVBPI database user that you created during the installation.

Complete the following steps to make the changes to the database password:

1. Start the Administration Console as described in section Administration Console Description on page 60.
2. Stop all the OVBPI components using the `Stop All` button.
3. Select the `MS SQL Server Access` option on the OVBPI Administration Console and make the changes to the following fields:
   - `MSSQL login password`
   - `Confirm password`
4. Click `Apply` to apply the changes to the OVBPI configuration.
5. Return to the `Status` option and restart all the OVBPI components, using the `Start All` button.

You have now completed the tasks to change the database password details used by OVBPI components.

Component Configurations - Oracle Server Access

The Oracle Server Access parameters are available only when you select Oracle as the database from within the OVBPI installation procedure. If you select MS SQL Server as your database, the Oracle Server parameters are not available through the Administration Console.

The `Oracle Access` option enables you to manage the properties of the connection to the Oracle database. The database is used by the following components:

- Business Impact Engine
- Metric Engine
- Business Event Handler
- Notification Server
- Business Process Dashboard
- Intervention Client
- OVIS, where you have installed the OVBPI OVIS Custom Probes

These parameters are listed in Table 29.

Table 29 Oracle Server Access

Descriptive Parameter Name	Description
Oracle Server hostname	This is a non-modifiable field and shows the fully qualified domain name of the system where the Oracle Database is installed.
Oracle Server port number	This is a non-modifiable field and shows the port number that the Oracle Database is expecting connections on.
Oracle Server SID	This is a non-modifiable field and shows the name of the database sid configured for OVBPI.
Oracle Server login user	This is a non-modifiable field and shows the name of the user who has permission to read and write the OVBPI database files.

Table 29 Oracle Server Access

Descriptive Parameter Name	Description
Oracle Server login password	The password for the user with permission to read and write the OVBPI database files. This field can be used to resynchronize the OVBPI version of the password with the database password after the database password is changed using the database management tools.
Confirm password	Confirmation of the Oracle Server login password.

If you are changing the password for the database user with permission to read and write the OVBPI database tables, follow the instructions on section Changing the Password Details on page 125.

Changing the Password Details

To change the password for the OVBPI database user, complete the following steps:

1. Start the Administration Console as described in section Administration Console Description on page 60.
2. Stop all the OVBPI components using the `Stop All` button on the `Status` option.
3. Select the `Oracle Access` tab on the OVBPI Administration Console and make the changes to the following fields:
   – `Oracle Server login password`
   – `Confirm password`
4. Click `Apply` to apply the changes to the OVBPI configuration.
5. Return to the `Status` option and restart all the OVBPI components, using the `Start All` button.

You have now completed the tasks to change the database password details used by OVBPI components.

Component Configurations - Port Numbers

This section describes the port numbers used by the OVBPI components. You might need to modify these if there is a port number clash, or if you have specific requirements with personal firewalls or other security processes.

The port number parameters are listed in Table 30

Table 30 Port Numbers

Descriptive Parameter Name	Description
RMI Registry	The port number for SUN's implementation of the Java RMI Activation System Daemon (RMID), which is used by OVBPI as a reliable RMI registry. The RMI registry holds the directory of port numbers used by the OVBPI components to receive RMI requests. If this port number is in use by another application, the OVBPI Server components are unable to start.
Administration Console Server	The port number that the server component of the Administration Console uses to listen for incoming administration requests.
Engine SOAP Receiver	The port number used by the Business Impact Engine to receive incoming SOAP requests.
Engine XML Receiver	The port number used by the Business Impact Engine to receive XML requests.
Engine RMI Receiver	The port number used by the Business Impact Engine to receive incoming RMI requests.
Business Event Handler Source	The port number used by the Business Event Handler Engine Adaptor to receive incoming Events into the Business Event Handler.
Business Event Handler Control	The port number used to stop and start the Business Event Handler Engine Adaptor programmatically; see the *OpenView Business Process Insight Integration Training Guide - Business Events*, for more details.

Table 30 Port Numbers

Descriptive Parameter Name	Description
Model Repository	The port number used by the Model Repository to listen for incoming RMI requests.
OpenView Operations Adaptor Request	The port number used by the OVBPI OpenView Operations Adaptor to receive incoming RMI requests from the Business Impact Engine. Note that the port numbers entered on the OVBPI Server system for the OpenView Operations Request must match the numbers entered for the OpenView Operations port numbers on the system where the adaptor is installed.
OpenView Operations Adaptor Reply	The port number used by the OVBPI Operations Adaptor when sending OVO responses related to the change in status of OVO Services, and for the Business Impact Engine to listen for responses. Note that the port numbers entered on the OVBPI Server system for the OpenView Operations Reply must match the numbers entered for the OpenView Operations port numbers on the system where the adaptor is installed.
Servlet Engine HTTP	The port number that the Tomcat Web Server uses to receive HTTP requests.
Servlet Engine Shutdown	The port number used by the Tomcat Servlet Engine to shutdown.
Servlet Engine AJP 1.3	The AJP 13 protocol is a packet-based protocol that allows a Web server to communicate with the Tomcat JSP/Servlet Container over a TCP connection.
Notification Admin	The port number that the Notification Admin uses to receive requests.
Servlet Engine Admin	The port number used by the Tomcat Servlet Engine administration tools.

To change the `OpenView Operations Adaptor Request` or `OpenView Operations Adaptor Reply` port numbers; refer to section OpenView Operations Adaptor Port Numbers on page 128.

For all other port number changes, refer to section Modifying OVBPI Port Numbers on page 129.

OpenView Operations Adaptor Port Numbers

The OVBPI OpenView Operations Adaptor comprises a client and a server component. The client component is embedded in the Model Repository and the Business Impact Engine components and the server component runs on the same system as OpenView Operations Adaptor.

If you need to alter the port numbers used by the adaptor server or client, complete the following steps:

1. Start the Administration Console on the system where the OVBPI Server is installed as described in section Administration Console Description on page 60.
2. Stop the OVBPI components using the `Stop All` button on the `Status` option.
3. Start the OVBPI Administration Console on the system where the OVBPI OpenView Operations Adaptor Server is installed.

   If the adaptor is installed on HP-UX, use the following command to start it:

   ```
   OVBPI-install-dir/bin/biaadmin.sh
   ```
4. Stop the `OpenView Operations Adaptor Server` component using the `Stop` button.
5. On the system, where the OVBPI OpenView Operations Adaptor is installed, select the `Port Numbers` option from the Administration Console.
6. Modify one or both of the following port numbers as appropriate:
   - `OpenView Operations Adaptor Request`
   - `OpenView Operations Adaptor Reply`
7. On the system where the OVBPI server is installed, select the `Port Numbers` option from the OVBPI Administration Console.

8. Modify one, or both of the following port numbers as appropriate:

   – `OpenView Operations Adaptor Request`

   – `OpenView Operations Adaptor Reply`

   Note that the port numbers entered on the OVBPI Server system for the OpenView Operations Request and Reply must match the numbers entered for the OpenView Operations port numbers on the system where the adaptor is installed.

9. Use the OVBPI Administration Console to restart the OVBPI component on the system where OpenView Operations adaptor is installed.

10. Use the OVBPI Administration Console to restart the OVBPI components on the system where the OVBPI Server is installed.

You have now completed the tasks to reconfigure the port numbers for the OVBPI OpenView Operations Adaptor.

Modifying OVBPI Port Numbers

If you want to modify the OpenView Operations Adaptor port numbers, refer to section OpenView Operations Adaptor Port Numbers on page 128.

The OVBPI components use a number of different port numbers for connections. There should not be any need to modify these port numbers unless you know that other applications are already using them, or if you are using personal firewall software that filters incoming connections based on port number. You must stop all the OVBPI components before you can modify the OVBPI port numbers.

To change the port number for the OVBPI Server components, complete the following steps:

1. Start the Administration Console as described in section Administration Console Description on page 60.

2. Using the `Status` option, stop the OVBPI components using the `Stop All` button.

   You cannot modify any of the port numbers until you have stopped the OVBPI components.

3. Select the `Port Numbers` option in the Navigator pane.

4. Make the changes to the port numbers that you want to modify.

5. Click `Apply` to apply your modifications.
6. Restart, using `Start All`, the OVBPI components that you previously stopped.

The port number changes have now been applied.

Component Configurations - Logging

This section describes how you can modify the log levels for the OVBPI components. OVBPI includes a number of third-party products, which use their own logging mechanisms, for example:

- the Business Event Handler component, which is based on the openadaptor framework, uses Log4j as its logging mechanism.
- Hibernate, which is used by the Business Impact Engine and other OVBPI Server components, uses Jakarta Apache Commons logging.

All the HP owned OVBPI components are based on the OpenView mechanism for logging, which is Java J2SDK version 1.4 logging.

Log4j and Java J2SDK version 1.4 logging are two alternative implementations and Apache Commons is an layer above a specific logging implementation. In the case of OVBPI, Apache Logging is a layer over the Java J2SDK version 1.4 logging implementation.

The Logging option enables you to set logging levels for the OVBPI components and for the Business Event Handler, which is based on openadaptor. There is no mechanism for setting the logging levels for Hibernate using the Administration Console, as the logging for the components using Hibernate is usually sufficient.

The logging parameters that can be set through the Logging option are listed in Table 31. For your OVBPI implementation, you see only those logging parameters that relate to the components that you have installed.

Table 31 Logging

Descriptive Parameter Name	Description
Log Viewer Application	
Path for the log viewer application	The full path name for the application that is being used to display the log files. If the application is defined as being on your `PATH`, you can enter the application name without its full path name.
OpenView Logging	

Table 31 Logging

Descriptive Parameter Name	Description
The Administration Console log level	The log level set for the Administration Console.
The Administration Console Server log level	The log level set for the Administration Console Server.
The Metric Engine log level	The log level set for the Metric Engine
The Engine log level	The log level set for the Business Impact Engine.
The Model Repository log level	The log level set for the Model Repository.
The Notification Server log level	The log level set for the Notification Server
The OVBPI OpenView Operations Adaptor log level[1]	The log level set for the OpenView Operations Adaptor.
The Servlet Engine log level	The log level set for the Tomcat Servlet Engine.
Log4j Logging	
The Business Event Handler log level	The logging level set for the Business Event Handler.

1. Available only when the OVBPI OpenView Operations Adaptor is installed.

The following are the logging levels that you can set for the `OpenView Logging` in ascending level of log detail provided:

1. `Info`, which reports runtime informational messages and is the default level of logging.
2. `Fine`, which provides more detail and is used when debugging or diagnosing problems.
3. `Finer`, which provides the greatest detail and is used for tracing and debugging low level problems.

The following are the logging levels that you can set for `Log4j Logging` in ascending level of detail:

1. `Info`, which reports at the informational level and is the default level of logging.
2. `Trace`, which should be used for debugging purposes.

When setting logging levels to report more detail, make sure that these are temporary settings as the detail logged can consume a significant amount of disk space and impact the performance of the system where OVBPI is installed.

Modifying the Log File Parameter Values

To modify the log levels for the OVBPI server components complete the following steps:

1. Start the Administration Console as described in section Administration Console Description on page 60.
2. Stop the server components using the `Stop All` button.
3. Select the `Logging` option in the Navigator pane.
4. Make the changes to the log levels that you want to modify.
5. Click `Apply` to apply your modifications.
6. Restart the OVBPI components that you previously stopped.

The logging level changes have now been applied.

The log levels set when OVBPI is installed are suitable for most error reporting. If you need to increase the granularity of the logging information provided, make sure that you set the log levels for a temporary period only. When set the `Fine` and `Finer` logging levels, the log files become large very quickly and can have a detrimental effect on your system's performance.

License Management

You use the License Management software (Autopass) to manage the HP OpenView software license keys. When you purchase an HP OpenView product, for example OVBPI, you receive an Entitlement Certificate. This Entitlement Certificate contains information that you need to retrieve and install OVBPI license keys.

The License Management utility is not an OVBPI component. OVBPI provides access to the utility to make it easier for you to request and install the OVBPI license keys. You can also request your license keys using email or facsimile (fax). Requesting license keys is described as part of the License Key installation in the License Management online help.

The License Management utility provides a number of features, including:

- License key installation
- License key reporting
- License key backup
- License key removal
- License key recovery

Details of these features and how to use them are provided in the License Management online help.

To install your OVBPI license keys complete the following steps:

1. Make sure that you have your Entitlement Certificates, which contain the order numbers that you need to retrieve your license keys.
2. Open the OVBPI Administration Console.
3. Select `License Manager` from the `File` menu on the Administration Console to start the Autopass License Management software.

   The License Management utility opens at the screen to `Retrieve/Install License Key`.
4. Follow the on-screen instructions and the instructions in the License Management online help to install the OVBPI license keys.
5. Select the `Report License Key` option from the License Management Navigation window to check that the OVBPI license keys are successfully installed.

When the keys are successfully installed, you should see the OVBPI product number with an `Expiration Date` of `Forever`.

If you have problems retrieving or installing the license keys, contact your local support representative. The Administration Console provides access to the Support Web site from the `eCare Support` option on the `Help` menu.

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 139.
- 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 144.

- 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 144. The violations can be reported to the business manager through email using the Notification Server.

  Refer to Chapter 9, Notification Server Configuration 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 145.
- Configuring the OVBPI custom probes for multiple OVBPI Servers; see section Configuring Probes for Multiple OVBPI Servers on page 163

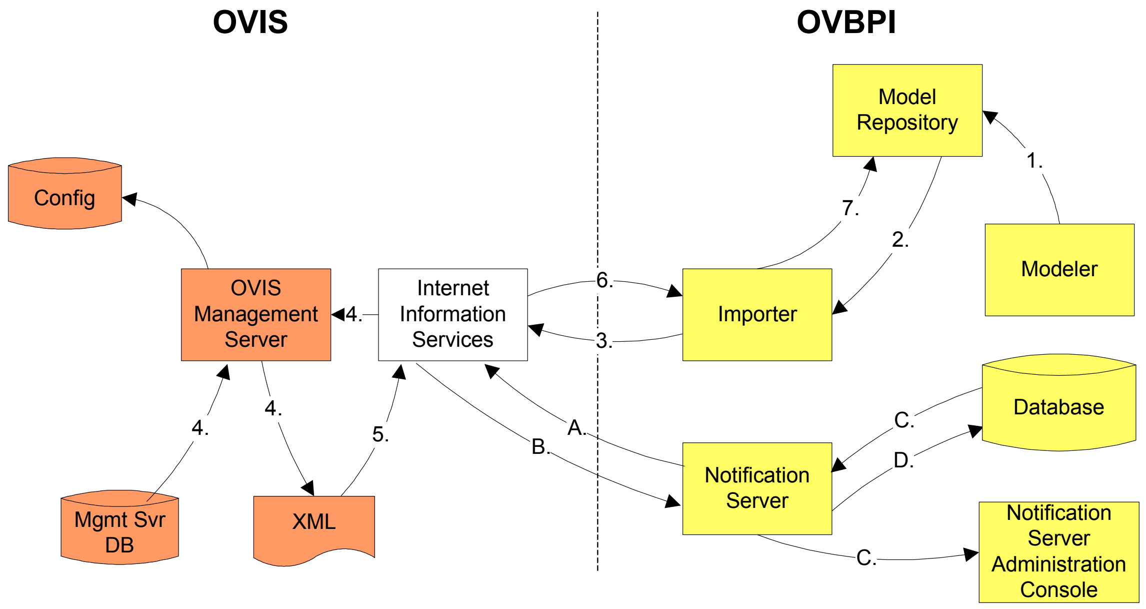
OVBPI and OVIS Integration

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 139 and in section OVBPI and OVIS Run-Time Integration on page 141.

OVBPI and OVIS Design-Time Integration

Figure 17 on page 139 shows the OVBPI and OVIS design-time configuration.

Figure 17 OVBPI and OVIS Design-Time Integration



There are two design-time scenarios shown in Figure 17 on page 139:

- 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 import 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 Chapter 3, OVBPI Component Administration 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.

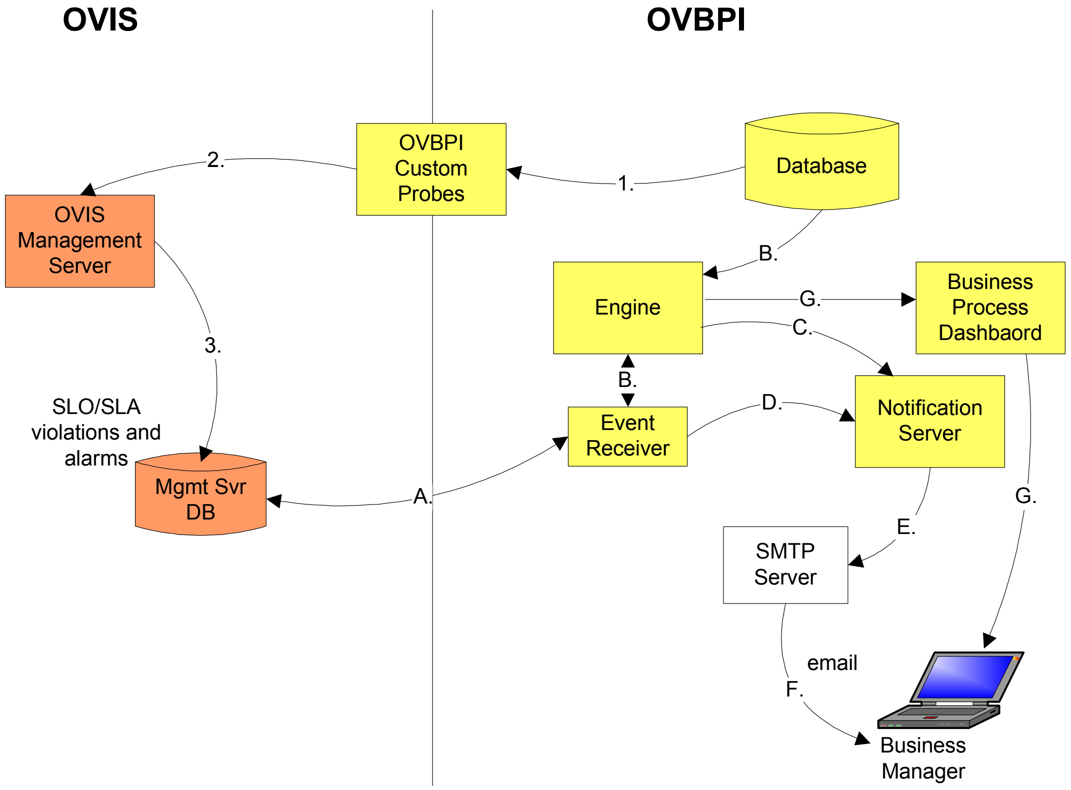
Step D: Details of the user subscriptions are stored in the OVBPI database.

Email notifications are delivered as described in the run-time integration; see section OVBPI and OVIS Run-Time Integration on page 141.

OVBPI and OVIS Run-Time Integration

Figure 18 on page 141 shows the OVBPI and OVIS run-time configuration.

Figure 18 OVBPI and OVIS Run-Time Integration



There are three run-time integration scenarios shown in Figure 18.

- 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 18 on page 141:

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 18 on page 141:

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 18 on page 141:

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 147 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 148.

   — to monitor business process metric durations, refer to section Flow Metrics (C_OVBPI_METRIC) on page 150.

   — to monitor flow instances for a specific node, refer to section Node Instances (C_OVBPI_NODE) on page 152.

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 32 for the Service Target named `C_OVBPI_FLOW`.

Table 32 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 163.
`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 163. The default value for the `OVBPI_Identifier` is `OVBPI`.
`OVBPI_DB_User_Name`[1]	The OVBPI database username that you provided for OVBPI during the OVBPI installation process.

Table 32 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.

1. 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 33 for the Service Target named `C_OVBPI_METRIC`.

Table 33 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 163.
`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 163. The default value for the `OVBPI_Identifier` is `OVBPI`.
`OVBPI_DB_User_Name`[1]	The OVBPI database user name that you specified for OVBPI during the OVBPI installation process.

Table 33 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`.

1. 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 34 for the Service Target named C_OVBPI_NODE_PROBE.

Table 34 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 163.
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 163. The default value for the OVBPI_Identifier is OVBPI.
OVBPI_DB_User_Name[1]	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 34 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.

1. 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 147.

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 36 on page 158 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 156, for the OVBPI `C_OVBPI_FLOW_PROBE` Monitored Service.
- C_OVBPI_METRIC_PROBE Objective Information on page 158, for the OVBPI `C_OVBPI_METRIC_PROBE` Monitored Service.
- C_OVBPI_NODE_PROBE Objective Information on page 159, 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 35 on page 156. 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 35 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 35 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 36 on page 158. You can define more than one Alarm, Service Level Objective and associated OVIS metric.

Table 36 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 37 on page 159. 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 37 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 37 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=hostname
```

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 always the ODBC driver name for a Microsoft SQL Server database.
- *`schema-name`* is the name of the database schema that you configured for the OVBPI Server on the hostname specified by *`hostname`*.
- *`hostname`* is the fully qualified hostname of the system where the database used by the OVBPI Server is installed.

```
[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 always the ODBC driver name for an Oracle database.
- `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=localhost
```

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, adaptors 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. For information about the predefined flows and adaptors 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 Chapter 9, Notification Server Configuration and the *OpenView Business Process Insight Concepts and Modeling Flows*.

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 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.

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\ovbpidashboard2-0\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.

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 168 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 section Component Configurations - Model Repository on page 110.

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 167.

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 is created, and subsequently mapped to an OVSD service, 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 Modeler Administration

This chapter describes the administration tasks related to the OVBPI Modeler.

The chapter describes specific tasks relating to the management of your definitions. For details of creating definitions and using the OVBPI Modeler refer to the *OpenView Business Process Insight Integration Training Guide - Modeling Flows* and the *OpenView Business Process Insight Concepts and Modeling Flows*. You can also find information about using the OVBPI Modeler in the Modeler online help.

This chapter describes the following:

- How to start and stop the Modeler component; see section Starting and Stopping the Modeler on page 175.
- The configuration parameters relating to the OVBPI Modeler connection to the Repository Server; see section Changing the Details of the Repository Server on page 176.
- Deploying definitions after they have been defined; see section Deploying Flows and Dependencies on page 177.
- How to undeploy flow definitions from the Model Repository using the Modeler; see section Undeploying Definitions from the OVBPI Modeler on page 178.

- How to access the log files associated with the Modeler; see section OVBPI Modeler Log Files on Windows on page 180.
- Using the Modeler to export and import flow definitions; see section Exporting and Importing Flow Definitions on page 181.
- Copy properties between Data and Event definitions; see section Copying Properties Between Data and Event Definitions on page 183.
- Exporting definitions in order to recover unsaved changes; see section Exporting Definitions to Recover Unsaved Changes on page 184.

Starting and Stopping the Modeler

Before trying to start the OVBPI Modeler, make sure that the `Model Repository` component is started on the OVBPI Server system. Use the OVBPI Administration Console to start the Model Repository as described in section Starting and Stopping the OVBPI Server Components on page 66.

Starting the Modeler

To start the OVBPI Modeler, complete the following steps:

1. On the system where the Modeler is installed, start the OVBPI Modeler as follows:

   `Start|Programs|HP OpenView|Business Process Insight|Modeler`

   You are presented with an `OVBPI Modeler` dialog.

2. Enter details of the username and password to connect to the Model Repository. On an new installation, the username is `admin` and the password is `ovbpi`.

   The OVBPI Modeler uses the same username and password as is configured for the Repository Explorer. You can modify the Repository Explorer login credentials using the Tomcat Realm configuration. Appendix B, Servlet Container Authentication describes the `tomcat-users.xml` file that you can edit to change the login credentials for the Repository Explorer and therefore for the OVBPI Modeler.

   You can also modify the details of the Repository Server that you connect to when you start the OVBPI Modeler. This is described in section Changing the Details of the Repository Server on page 176.

3. Click `OK`, and the OVBPI Modeler opens.

Stopping the Modeler

To stop the OVBPI Modeler, select `Exit` from the `File` menu in the Modeler window. You are prompted to save any currently unsaved changes to the models that you have been editing.

Changing the Details of the Repository Server

Table 38 describes the OVBPI Modeler parameters that you can modify that relate to its connection to the Repository Server. These parameters are available from the `File|Options` menu in the OVBPI Modeler, or through the `Options...` button on the OVBPI Modeler login dialog box. You can modify the parameters on the login dialog only; the Server options within the `File|Options` menu are read-only.

Table 38 Repository Server Configuration Parameters

Descriptive Parameter Name	Description
Hostname	The hostname of the system where the OVBPI Server (and Repository Server) is installed.
Server Port Number	The port number used for the Java Remote Method Invocation (RMI) demon. The port number is required in order that Java processes can communicate with each other. You need to change this option if you modify the port number through the Administration Console on the OVBPI Server. The port number used by the OVBPI Server is shown within the OVBPI Administration Console under the `Port Number` option as `RMI Registry`.
Client Port Number	The port number used by the OVBPI Server when it needs to communicate with the Modeler. This port number can be zero (0), in which case, the OVBPI Server selects a port number at random; however, if you want to ensure the port number fits with your organization's firewall policies or other similar policies, you need to specify the port number explicitly.
Servlet Engine HTTP Port	The port number used for the Servlet Engine on the OVBPI Server system. You need to change this option if you modify the port number through the Administration Console on the OVBPI Server. The port number used by the OVBPI Server is shown within the Administration Console under the `Port Number` option as `Servlet Engine HTTP`.

Deploying Flows and Dependencies

This section explains how to deploy a Flow and any associated Data definitions, Event definitions and Services.

Before deployment, you have either created a design-time definition in the Model Repository that the Business Impact Engine has no knowledge of, or have undeployed a definition and made changes to it. When you deploy the definition to the Engine, the Engine then creates:

- a Java-compiled definition for the flow in its database
- database entries for the definition
- a mapping file

To deploy the Flow, Data, Event and Service definitions to the Business Impact Engine, complete the following steps:

1. Make sure the Modeler is started.
2. Select the Flow from the Navigator pane in the left-hand pane.
3. Select `File|Deploy` from the menu options.
4. You are prompted to save all currently unsaved changes, if you have not saved them already. Click `Yes`.

   The Modeler checks the state of all the definitions that you selected to deploy. A dialog box appears to prompt you to confirm that you want to deploy all currently undeployed changes to the business flow and all its dependencies. Click `Deploy` to confirm your request.

   The deployer shows its progress as it deploys the Services, Flows, Data and Event definitions.
5. Click `Close` when the Deployer indicates that it has finished deploying the definitions.

You have now deployed the flow and its dependencies.

Undeploying Definitions from the OVBPI Modeler

There are circumstances where you might want to undeploy one or more of your business flows. In addition, you might need to undeploy individual definitions from the OVBPI Model Repository; for example, you might need to undeploy Event definitions when an Event becomes out of date.

You can undeploy Flows, Data and Event definitions, you cannot undeploy services. This is because services are identifiers for operational services that exist in other applications, for example OVIS. The deployment status for services is marked, in the Modeler interface, as `Deployed or Not Applicable`.

In the case of OpenView Operations services, a warning is shown if there is no equivalently named service in OpenView Operations. The warning is removed when the service is created in OpenView Operations and the OVBPI Modeler has successfully imported the revised service definitions.

➤ If you undeploy a definition, existing instances using the definitions are still running in the Business Impact Engine continue to run until completion, or until the Instance Cleaner deleted them. New instances of any undeployed definitions are no longer created. The definition is removed by the Model Cleaner as described in section Modifying the Engine Model Cleaner Settings on page 83.

You can undeploy a definition that is referenced by another definition, so you need to make sure that you maintain consistency within your business flows.

To undeploy a Flow, Data or Event definition, complete the following steps:

1. Start the OVBPI Modeler as follows:

   `Start|Programs|HP OpenView|Business Process Insight|Modeler`

   You are presented with an `OVBPI Modeler` dialog, where you enter details of the location of the Model Repository.

2. Enter details of the username and password to connect to the Model Repository. On an new installation, the username is `admin` and the password is `ovbpi`.

   You can modify these credentials using the Tomcat Realm configuration. Appendix B, Servlet Container Authentication describes the `tomcat-users.xml` file that you can edit to change the login credentials for the Model Repository.

   You can also modify the details of the Repository Server that you connect to when you start the OVBPI Modeler. This is described in section Changing the Details of the Repository Server on page 176.

3. Click `OK`, and the OVBPI Modeler opens.

4. Select the definition that you want to undeploy from the Navigator pane in the OVBPI Modeler.

5. Select `Undeploy` from the `File` menu and follow the instructions provided.

   The status of the definition changes from `Yes` to `No, but can be deployed`.

The `Summary` Window in the OVBPI Modeler now shows the revised status of the definition.

If you have modified a definition, it is no longer seen as being deployed within the OVBPI Modeler. In this case, you are unable to undeploy it. If you want to know what the deployed status of definitions is (as opposed to a development status), use the Intervention Client to show you the status of the definitions. The Intervention Client is described in the *OpenView Business Process Insight Administration Guide*.

You can also view your definitions using the Repository Explorer, see Chapter 8, Repository Explorer.

OVBPI Modeler Log Files on Windows

The OVBPI Modeler log files are called `bia_modeller`*`n_n`*`.log` and are located in the following directory on the Windows system where the OVBPI Modeler is installed:

`OVBPI-install-dir``\data\log`

The log file is a text file that you can open using a text editor. Using the log files to solve problems with your OVBPI system is described in the *OpenView Business Process Insight Problem Solving Guide*.

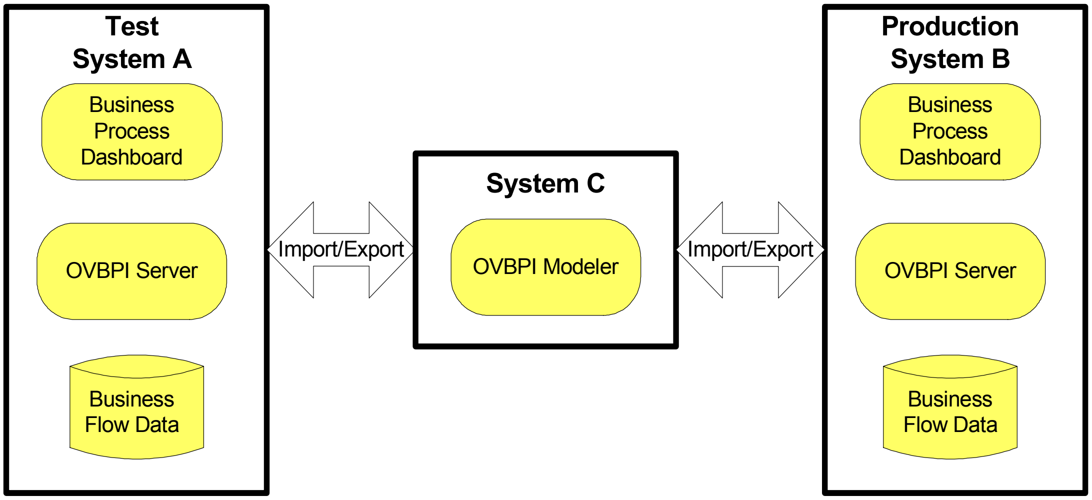
There are no logging settings specifically for the Modeler. The Modeler uses `The Model Repository log level` parameter value for its logging levels. Refer to section Component Configurations - Logging on page 131 for details of setting the logging levels for the Model Repository.

Exporting and Importing Flow Definitions

The OVBPI Modeler has an option to enable you to export and import business flows, including their dependencies (Data, Service and Event definitions) to a .zip file that contains XML files. This enables you to create business flows using one Modeler for testing and then move the flows to a production system when they are fully verified. You can also export and import any business process metric definitions that you have defined for your flows as described in Exporting and Importing Metric Definitions on page 187.

The following figure shows an example scenario for using the import and export function of the OVBPI Modeler.

Figure 19 Remote OVBPI Modeler



You use the `File|Import Definitions...` and `File|Export Definition...` menu options from the OVBPI Modeler to export and import your business Flow definitions.

Make sure that you have a naming policy for Flow, Data and Event definitions across your organization to minimize the possibility of a name clash when the business flows are imported for deployment to the Business Impact Engine. You have the option to rename your definitions when you import them; however, you might prefer to have a standard naming scheme.

Exporting Definitions

Use the `File|Export Definition...` option to export the business flows to an XML .zip archive file. When you have exported the file, you can copy the file to the system where the OVBPI Modeler that you want to import the file to is installed. Finally, use the Import option to import the Flow definitions as described in section Importing Definitions on page 182.

Importing Definitions

If you have previously exported a Flow definition and you now want to import it into the OVBPI Modeler, complete the following steps:

1. Select the following option from the `File` menu:

   `Import Definitions...`

2. Navigate, or browse, to the location of the previously exported definition file and click `Open`.

3. Click `Next` to move to the next option. You are presented with a dialog that lists the content of the definition file that you are importing.

   From this dialog, you can:

   — import a listed definition and overwrite any existing definition of the same type that has the same name. If you provide a new name for the definition, the definition is overwritten and renamed.

   — create a new copy of a listed definition. For this option, you provide a new name for the definition in the `New Name` column.

   — choose not to import a listed definition.

   In the case of Service definitions, existing services are referenced automatically.

   When you have tailored your import requirements, click `Import`.

4. Click `Close` to complete the import.

When the Flow definitions are successfully imported they are shown as being undeployed in the OVBPI Modeler. You now need to deploy the imported Flows and their dependencies to the target OVBPI Server system.

Copying Properties Between Data and Event Definitions

Using this option enables you to create properties in your Data definition that match the properties of an Event definition that it subscribes to. Similarly, it can be used to create properties of an Event definition that match those in a Data definition.

Using the copy function saves the need to define identical properties for your Data and Event definitions multiple times. To copy properties from one definition to another definition, complete the following steps:

1. Open the definition whose properties you want to define.

   The properties that you select in the following steps are copied to this definition.

2. Select the Properties tab for the definition.
3. Select the Copy Properties option from the Tools menu.

   You are presented with a Copy Properties dialog that has a tree definition of all the Data and Event definitions currently defined.

4. Select the definition whose properties you want to copy.

   Note that you are not able to copy a property that already exists in the selected definition.

5. Select the properties within the definition that you want to copy. You can use the `Select All` option to select all the properties.
6. Click `OK` and the selected properties are copies to the definition that you opened in step 1.

You have now copied properties from one definition to another definition.

Exporting Definitions to Recover Unsaved Changes

If the OVBPI Modeler loses its connection to the Repository Server for any reason, for example, the OVBPI Server has been shutdown in order to reconfigure a component, you cannot continue and modify your Flow definitions.

If you have made changes to your definitions, but not saved the changes, you have the opportunity to export these changes to a recovery file before closing the OVBPI Modeler.

➤ If you close the OVBPI Modeler without exporting your unsaved changes, the changes are lost.

As soon as the OVBPI Modeler realizes that it has lost its connection to the Repository Server, it presents a dialog for you to enter the name of the file where you want to save your modifications. The next time that you start the OVBPI Modeler and the Modeler successfully connects to the Repository Server, you are presented with a dialog that enables you to select the file where you saved your changes in order to import them back into the Repository Server.

If the OVBPI Modeler loses its connection to the Repository Server, and all your definitions are up to date, you are informed that the OVBPI Modeler must exit, and that no data is lost.

When you have exported the unsaved modifications, the file that is created is the same as the files created using the `Export` option and you can import the file at a later time if required or preferred.

The file used to store the definitions for recovery purposed is not deleted after the recovery process is complete. It is therefore available in the future if you wanted to import your modifications again for any reason.

7

OVBPI Metric Definer Administration

This chapter describes the administration tasks related to the OVBPI Metric Definer. Using the Metric definer is described in *OpenView Business Process Insight Concepts and Modeling Flows* and the *Integration Training Guide - Defining Business Metrics* and the *Concepts and Modeling Flows*.

Managing the Metric Engine parameters is described in section Component Configurations - Metric Engine on page 94 of this guide and the architecture for the Metric Engine and Metric definer is described in Chapter 2, OVBPI Architecture of this guide.

The following sections describe how to view the log files associated with the business process metric definer, and the Metric Engine, plus how to import and export metric definitions.

Starting and stopping the Metric definer is described in Chapter 3, OVBPI Component Administration along with other OVBPI Server components; the Metric Engine and the Metric definer are components of the OVBPI Server.

OVBPI Metric Definer Log Files

There are two sets of log files that you need to be familiar with when managing your business process metrics:

- Metric Engine log files

  The Metric Engine log files are managed in the same way as other OVBPI Server components and are described in section Component Configurations - Logging on page 131.

- Metric definer log files

  The Metric definer logs messages to the OVBPI Servlet Engine log file. These messages in the Servlet Engine log file are prefixed with the string `Metric Definer`. The Servlet Engine is considered as a component of the OVBPI Server and section Component Configurations - Logging on page 131 describes the Servlet Engine log file.

Exporting and Importing Metric Definitions

The OVBPI Metric definer has options that enable you to export and subsequently import the business process metrics, metric thresholds and filters defined for a selected business flow.

These definitions can be exported to a `.zip` archive file, which contains XML files describing the definitions. The `.zip` archive contains a copy of the metric definition and some information about the parent flow definition; this is sufficient information to uniquely identify the Flow definition that the business process metric applies to. The `.zip` archive file does not contain any information relating to files that you might have created for custom business process metrics; you need to separately manage any SQL you have created for your custom metric definitions.

Be aware that the `.zip` archive created from the Metric definer is not the same as the `.zip` archive created from the Modeler. The `.zip` archive from the Metric definer, contains sufficient information about the business flow to enable the Metric definer to import the metric; it does not contain a full definition of the business flow. You cannot interchange the `.zip` archives between the two applications, you must import the archive created from the appropriate application.

You might want to export a business process metric definition to enable you to create business process metrics for testing and then move these metrics to a production system when they are fully verified.

➤ You cannot import a metric definition until the flow definition that the metric applies to is deployed to the Business Impact Engine. Refer to section Exporting and Importing Flow Definitions on page 181 for details of exporting and importing flow definitions.

Exporting a Business Process Metric

To export one or more business process metrics, use the `Export` or `Export All` options from the `File` menu in the Metric definer Navigation frame.

When you select a business process metric to export, all the associated metric threshold and filters are also exported. The export is based around the identifier of the business flow that the business process metric is linked to. All the information, including metric thresholds and filters, is exported to XML files, which are then packaged into an overall `.zip` archive file. The `.zip` file can then be copied to the destination system in order to be imported.

When you export a business process metric, you also need to make a copy of any additional files that you might have created that are associated with the business process metric; for example, you might have defined a custom metric. In the case of a custom metric, you need to make sure that you have a copy of any SQL files that you have created for the custom metric.

The name of the exported file is based on the Flow name, machine name and time that the metric is exported. This is in order to provide a unique name for the `.zip` archive file. If any of the elements of this constructed file name contain non-ASCII characters, the elements are excluded from the file name. This is because Internet Explorer allows only `.zip` archive file names that contain ASCII characters.

Importing a Business Process Metric

Before importing a business process metric, you must have already imported and deployed the business flow that the business process metric is linked to.

In addition, if you have defined custom metrics, and you are moving the definitions to a new machine, make sure that you also copy any files that you have created for the custom metric to the new machine. For example, you might have created SQL files for your custom metric and this SQL file is not exported as part of the `.zip` file. If you do not copy the files relating to any custom metrics before importing the Metric definition, the import fails.

To import the business process metric, complete the following steps:

1. Select the Flow that you want to import metric definitions for from the left-hand Navigation frame.
2. Select the following option from the `File` menu on the Metric definer navigation frame:

   `Import`

   The right-hand pane provides a browse button that you can use to navigate to the file where the file that you want to import is located.
3. Select the file that you want to import and click `Open`.
4. Click the `Import Definitions` button.

   The Metric definer provides a summary of the metric definitions contained in the zip file and how they will be applied to the flows that are currently defined on your system. You need to check that the metric definitions in the zip file are being applied to the correct flow definitions before importing the file. As an example, you need to make sure that you have not renamed a flow since a metric definition was exported.
5. Select the checkbox next to each set of business process metrics that you want to import from the zip archive file.
6. Click `Import Definitions`.

   The Metric definer displays a feedback message indicating whether or not it successfully imported the selected metric definition.
7. Click `OK` and the Metric definer returns to the Metrics and Thresholds page for the flow.

If you import metric definitions which are named identically to metric definitions that are already defined within the Metric definer, they are imported and renamed (appended with a number), in order to make them unique. Threshold names are not changed as they need only be unique for the Metric definition that they apply to.

Repository Explorer

This chapter describes the OVBPI Repository Explorer. The Repository Explorer is a Web-based interface that enables you to browse and manage the contents of the Model Repository. The Model Repository holds the data for the business flows that you have defined using the OVBPI Modeler.

For details of creating definitions using the OVBPI Modeler, refer to the *OpenView Business Process Insight Integration Training Guide - Modeling Flows* and the *OpenView Business Process Insight Concepts and Modeling Flows*.

This chapter describes the following:

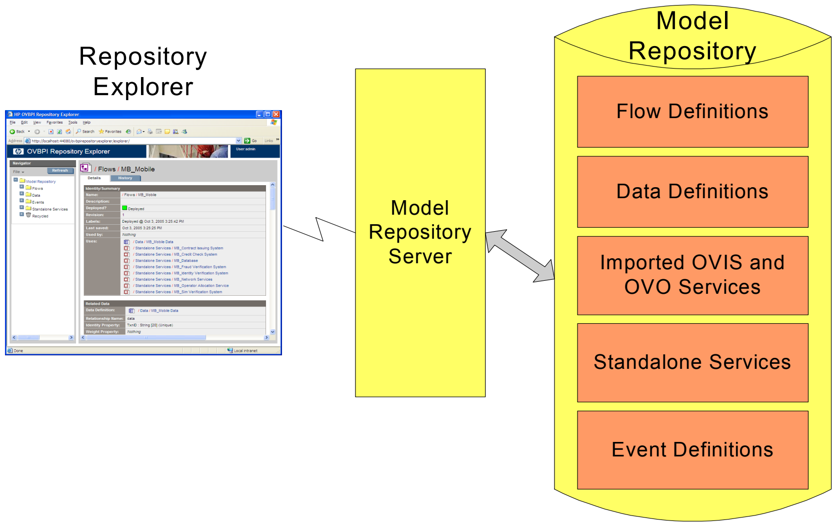
Repository Explorer Overview

Using the OVBPI Modeler, you can create, modify and subsequently deploy a definition. When you have deployed the definition, you might then go on and revise the definition and significantly change it. This means that, unless you have exported the version of the definition that you previously deployed, you no longer have a copy of it within the OVBPI Modeler. You can therefore be in the position of having a deployed flow, but no copy of the source for the flow.

The Repository Explorer enables you to always be able to access the source to versions of flows that you have created and deployed.

The Repository Explorer is a view on the Model Repository data held in the OVBPI database. Access to the data is provided through the Repository Server as shown in Figure 20.

Figure 20 Repository Explorer Architecture



Specifically, the Repository Explorer enables you to:

- browse the definitions that you have created.

  The OVBPI Modeler is optimized for editing and not for browsing and therefore does not present the information relating to Flow, Data, Event and Service definitions in an easy-to-view way. In addition, the Modeler does not enable you to access data relating to superseded versions of definitions. Using the Repository Explorer, you can easily browse the current and earlier revisions of a definition to quickly see how they were defined.

- export the latest version of a definition.

  You might want to do this in order to take a copy of the latest revision of the definitions. The latest version of the definition is exported to a `.zip` file. Exporting the definitions through the Repository Explorer, or the OVBPI Modeler, also ensures that the definitions are consistent within themselves as the definition and all its dependencies are exported.

- export the latest version of all definitions.

  You might want to do this in order to move the definitions from one OVBPI Server to another OVBPI Server, for example, from a development system to a staging system. The latest versions of all the definitions within the Model Repository are exported to a `.zip` file, including all the dependencies.

- export a superseded version of a definition.

  You might want to do this if you have not saved a particular version of a definition, when using the OVBPI Modeler to edit it, and you need to keep a record of it.

- remove definitions.

  You can delete any definition using the Repository Explorer, provided it is not the current, or any superseded, version of the definition that has instances currently running in the Business Impact Engine.

- restore a definition that has been deleted using the OVBPI Modeler or Repository Explorer.

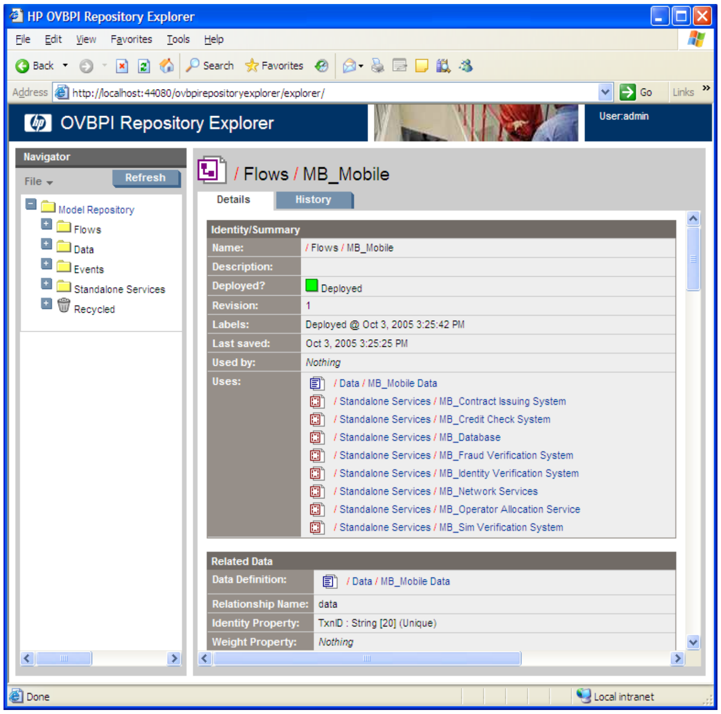
- permanently remove deleted definitions from the recycle folder.

- print definitions.

  There is a print option within the Repository Explorer that enables you to print the information listed in the `Details` or `History` views.

Figure 21 shows an example of the layout of the OVBPI Repository Explorer when you open it in a Browser Window.

Figure 21 Repository Explorer



Identity/Summary	
Name:	/ Flows / MB_Mobile
Description:	
Deployed?	🟩 Deployed
Revision:	1
Labels:	Deployed @ Oct 3, 2005 3:25:42 PM
Last saved:	Oct 3, 2005 3:25:25 PM
Used by:	*Nothing*
Uses:	/ Data / MB_Mobile Data / Standalone Services / MB_Contract Issuing System / Standalone Services / MB_Credit Check System / Standalone Services / MB_Database / Standalone Services / MB_Fraud Verification System / Standalone Services / MB_Identity Verification System / Standalone Services / MB_Network Services / Standalone Services / MB_Operator Allocation Service / Standalone Services / MB_Sim Verification System

Related Data	
Data Definition:	/ Data / MB_Mobile Data
Relationship Name:	data
Identity Property:	TxnID : String [20] (Unique)
Weight Property:	*Nothing*

The Repository Explorer comprises:

- A left-hand Navigator frame that lists the deployed Flow, Data, Event and Service definitions. The Navigator frame also contains a `Recycled` folder where definitions that have been deleted from the OVBPI Modeler are listed.

  There are also a number of menu options on the Navigator frame.

- A right-hand frame that displays the appropriate details of the definition that you have selected in the Navigator frame. The right-hand frame includes tabbed pages for `Details` and `History`.

  The right-hand frame also contains a print icon, which enables you to print the contents of the right-hand pane.

The features offered by the Repository Explorer are described in more detail in the following sections.

Navigator Frame

The Navigator frame lists all the definitions that have been created in the Model Repository, using the OVBPI Modeler; the deployment status of each definition is also displayed in the Navigator frame. The Repository Explorer uses the same icons as are used within the OVBPI Modeler to show whether a definition is deployed or undeployed.

The following are the menu options available within the Navigator frame:

- `File`
- `Refresh`

  Click the `Refresh` button to refresh the Navigator frame to be up to date with the latest definitions that have been recently entered into the Model Repository using the OVBPI Modeler.

The following are the options are accessible from the `File` menu:

- `Export`

  Enables you to export the selected definition to a zip file; see section Exporting Flow Definitions on page 202.

- `Export All`

  Enables you to export the latest revision of all the definitions in the Navigator frame to a zip file; see section Exporting Flow Definitions on page 202.

- `Restore`

  This option is available when you select a definition from the `Recycled` folder. It restores the definition back to its original location, in the Model Repository, at the point where it was deleted in the OVBPI Modeler; see section Restoring a Definition on page 205.

The Navigator frame also includes a `Recycled` folder. Under the `Recycled` folder is a list of definitions that have been deleted from the OVBPI Modeler. When you delete a definition using the OVBPI Modeler the definition is moved to this `Recycled` folder, which is accessible only from the Repository Explorer.

- `Cleanup Recycled`

  This option deletes all definitions in the `Recycled` folder and undeploys definitions that are deployed. If a definition is still in use by the Business Impact Engine, it is not deleted or undeployed. When all instances of these definitions have completed, these entries can then also be removed using the `Cleanup Recycled` option and the entries no longer appear in the interfaces. The instances of definitions are deleted by the Business Impact Engine using the Engine Instance Cleaner.

When you select a definition in the Navigator frame, the right-hand frame opens at the `Details` tab; see section Details Tab on page 196. You also have the option of selecting the `History` tab from the right-hand frame; see section History Tab on page 198.

Details Tab

The Details tab provides a page listing the details of the definition that you have selected in the Navigator frame. Much of the information listed under the `Details` tab can be determined from the OVBPI Modeler; however, the Repository Explorer provides the information in a structured form, which makes it easier to reference and print. The information listed under the `Details` tab varies according to type of definition selected in the Navigator frame:

- Flows
- Data
- Events
- OVIS Services
- OVSN Services
- Standalone Services

The following is an overview of all the sections for all the definitions:

- Identity/Summary

  This section appears for all the definitions and lists an overview, or summary, of the definition.This includes information that you have entered relating to the definition using the OVBPI Modeler, and information relating to the revision of the definition.

  This section also lists the revision number of the definition and details of the other definitions that the selected definition uses and is used by.

- Related Data

  This section appears only for a Flow definition and lists details of the Data definition that has been defined as the Related Data definition for the selected Flow definition.

- Flow Diagram

  This section appears only for a Flow definition and shows the flow diagram of the selected flow. It is the same flow diagram as is shown through the OVBPI Modeler.

- Nodes

  This section appears only for a Flow definition and lists details of the individual Nodes that make up the Flow definition, including:

  — Name of the Node

  — Type of Node (Start, End or Activity)

  — Progression Rules (Start and Complete conditions)

  — Details of any operational services that the Node relies on.

- Checked Arcs

  This section appears only for a Flow definition and lists details of the arcs that have been defined as Checked Arcs for the Flow definition.

- Properties

  This section appears for Data and Event definitions and lists details of the Properties that have been defined for the selected definition.

- Subscriptions

  This section appears only for a Data definition and lists details of the Event subscriptions related to the Data definition.

- ToDo List

  This section appears for all the definitions and lists the content of the current to-do list for the selected definition.

History Tab

The History tab provides a page tabulating the details of the revision history related to the selected definition in the Revision History table.

These details include:

- Revision number
- Deployment status
- Name
- Date the revision was created
- Description (from the Description field in the definition)
- Label, which is the date and time that the revision was deployed.

Each entry in the table has a check box which you can select (and clear). When you select the check box for one of the definitions in the table, the options available depend on how many definitions are selected. The Export option is available for single options only. If you select more than one definition, or do not select any definitions, the Export option is not available.

If you want to export a definition on the Revision History table, select the definition, click the `Export` button and then follow the instructions that are presented to you.

There is also a delete option available for each revision of the definition. You cannot delete a definition that is currently being processed by the Business Impact Engine, if you do try to delete such a definition, the Repository Explorer issues a warning message and does not allow the delete action to progress.

You can select any version of a definition to delete, including the current version of a definition. In the case of the current definition, the Repository Explorer issues a warning message for you to confirm the delete action and then continues. Be aware that you cannot recover any definition that is deleted using the Repository Explorer.

Starting and Stopping the Repository Explorer

Before trying to start the OVBPI Repository Explorer, make sure that the `Model Repository` and `Servlet Engine` components are started on the OVBPI Server system. Use the OVBPI Administration Console to start them, as described in section Starting and Stopping the OVBPI Server Components on page 66.

Opening the Repository Explorer

To start the OVBPI Repository Explorer, complete the following steps:

1. Make sure that the `Model Repository` and `Servlet Engine` components are started.
2. Type the following URL into a Web Browser:

   `http://`*hostname*`:44080/ovbpirepositoryexplorer`

   where:

   — *`hostname`* is the fully qualified domain name of the system where the OVBPI Server is installed and running. You can use `localhost` as the hostname if you are starting the Repository Explorer on the system where the server components are installed and running.

   — `44080` is the port number for the Servlet Engine, identified by the `ServletEngine HTTP` port number. Use the port number configured for your system.

   You are prompted for a username and password.
3. Enter details of the username and password to connect to the Model Repository. On a new installation, the username is `admin` and the password is `ovbpi`.

   You can modify these credentials using the Tomcat Realm configuration. Appendix B, Servlet Container Authentication describes the `tomcat-users.xml` file that you can edit to change the login credentials for the Model Repository.
4. Click `OK`, and the OVBPI Repository Explorer opens.

Stopping the Repository Explorer

You close the Repository Explorer by closing the browser Window where the Explorer is running.

If you want to increase the level of security for the Repository Explorer pages, you can use the Web Server authentication mechanisms to lock the Web Browser screen after a certain length of time. Appendix B, Servlet Container Authentication provides details of the default authentication that is provided for the OVBPI Web interfaces.

Exporting Flow Definitions

The Repository Explorer can be used to export any revision of a definition from the Model Repository.

Note that this definition is compatible with the Modeler and not the Metric definer; it does not contain any business process metric information.

You might have multiple versions of a definition where you have deployed the Flow definition multiple times. You can also export the latest version of a definition using the OVBPI Modeler; however, you cannot export earlier revisions of a definition using the Modeler, you can do this only through the Repository Explorer.

In addition to exporting individual versions of a definition, you can also choose to export the latest revision of all definitions from the Model Repository; you might want to do this to move all the latest definitions to another OVBPI Server.

Exporting a Single Definition

To export a definition, complete the following steps:

1. Select the definition that you want to export from the left-hand Navigator pane.

   If you want to export a superseded version of the definition, continue at step 3; otherwise, continue at step 2.

2. Click the `Export` option from the `File` menu in the Navigation pane.

   Continue at step 6.

3. Select the `History` tab in the right-hand pane.
4. Select the revision of the definition that you want to export from the list.
5. Click the `Export` button in the right-hand pane.
6. The right-hand pane lists the definition that will be exported.
7. Click the `Download` button to continue to export the definition.

   You are presented with your browser's `File Download` dialog.

8. Select `Save`, to save the definition as a `.zip` file.

   You are presented with a `Save As` dialog, where you can specify a file name and directory location for the `.zip` file.

9. Enter the details of the file name and click `Save`.

   The file is saved and a `Download complete` dialog is displayed.

10. Click `Close` and the export is complete.

Exporting All Definitions

To export all definitions, complete the following steps:

1. Click the `Export All` Option from the `File` menu in the Navigation pane.
2. The right-hand pane lists the definitions that will be exported.
3. Click the `Download` button to continue to export the definition.

   You are presented with your browser's `File Download` dialog.

4. Select `Save`, to save the definition as a `.zip` file.

   You are presented with a `Save As` dialog, where you can specify a file name and directory location for the `.zip` file.

5. Enter the details of the file name and click `Save`.

   The file is saved and a `Download complete` dialog is displayed.

6. Click `Close` and the export is complete.

Undeploying a Flow Definition

You cannot use the Repository Explorer to undeploy a specific definition. You must use the OVBPI Modeler to do this. You can use the `Cleanup Recycled` option within the Repository Explorer to undeploy and delete all current revisions of definitions held in the `Recycled` folder; see section Navigator Frame on page 195.

Be aware that undeploying an Event definition, which is referenced by a Data definition that has current active instances, means that these data instances might not be able to progress. If this is the case, you can use the Intervention Client to delete the individual Data definition instances.

Restoring a Definition

The Repository Explorer can be used to restore definitions that have been deleted using the OVBPI Modeler. These definitions are listed in the Recycled folder on the Navigator frame.

You can access the `Restore` option from the `File` menu on the Navigator frame. The steps for restoring a previously deleted definition are as follows:

1. Select the definition that you want to restore from the Recycled folder.
2. Select the `Restore` option from the `File` menu in the Navigator frame.

   The definition is restored to the Model Repository and is removed from the Recycled folder in the Repository Explorer. The restored definition appears under the appropriate definition list in the Repository Explorer.

Repository Explorer Log Files

The Repository Explorer uses the Repository Server to access its data and the Repository Server logging is recorded under Model Repository in the Administration Console. Refer to section Component Configurations - Logging on page 131 for details of the logging settings that are used for OVBPI components.

In addition, you can find log messages for the Repository Explorer in the Servlet Engine log file.

9

Notification Server Configuration

This chapter describes the OVBPI Notification Server Web Administration Console and how to use it to configure subscriptions for alerts such as flow impact and metric threshold alerts.

The chapter also describes how to configure email templates and scripts, which can be used when notification alerts are received by the Notification Server.

The Notification Server is the component responsible for sending email alerts and OpenView Operations messages for events. These messages provide details of the flow impact, metric threshold and out-of-sequence events, plus SLO and SLA email violations. You can receive these alerts through your email client or through an OpenView client according to your requirements.

You also have the option to execute a script when an impact event is received; for example, the script might update a file, or send an SMS message.

The Notification component can be considered in two parts:

1. The server, which is responsible for receiving the impact, SLO and SLA events from the Business Impact Engine and converting them into the appropriate email messages and OpenView Messages for delivery. The architecture for the Notification Server is described in the Architecture chapter; refer to section Notification Server on page 42.
2. The Notification Server Web Administration Console, which enables you to create subscriptions for users to receive email notifications, SLOs and SLAs. It also enables you to create subscriptions for OpenView Operations messages and scripts. Using this Web Administration Console is the topic described in this chapter. The Web Administration Console uses Tomcat as its Servlet Engine to manage these Web pages. Tomcat is installed as part of the OVBPI Server.

The Notification Server components are shown in Figure 22 on page 210.

Specifically, this chapter describes:

- Configuring subscriber accounts for email alerts; see section Adding Users for Email Subscriptions on page 211. This includes:
  - — subscriber accounts to receive specific flow impact events that are generated by OVBPI.
  - — subscriber accounts to receive specific metric threshold events that are generated by OVBPI.
  - — subscriber accounts to receive specific out-of-sequence events that are generated by OVBPI.
  - — subscriber accounts to receive OVIS SLOs and SLA violation notifications.
- Configuring subscriptions to send messages to OpenView Operations; see section “Adding OpenView Operations Message Subscriptions” on page 219.

- Configuring the templates used by the Notification Server to send alerts.

  The Notification Server uses templates to format the email and OpenView Operations notifications. These templates contain instructions about how to transform the event information such as event name and event time into a format of your choice, subject to the restrictions of the transformation tool you are using. There are two types of templates that you can define to format a notification, these are Velocity templates and XSLT style sheets. Creating new templates for your user subscriptions is described in section Creating Notification Server Templates on page 225.

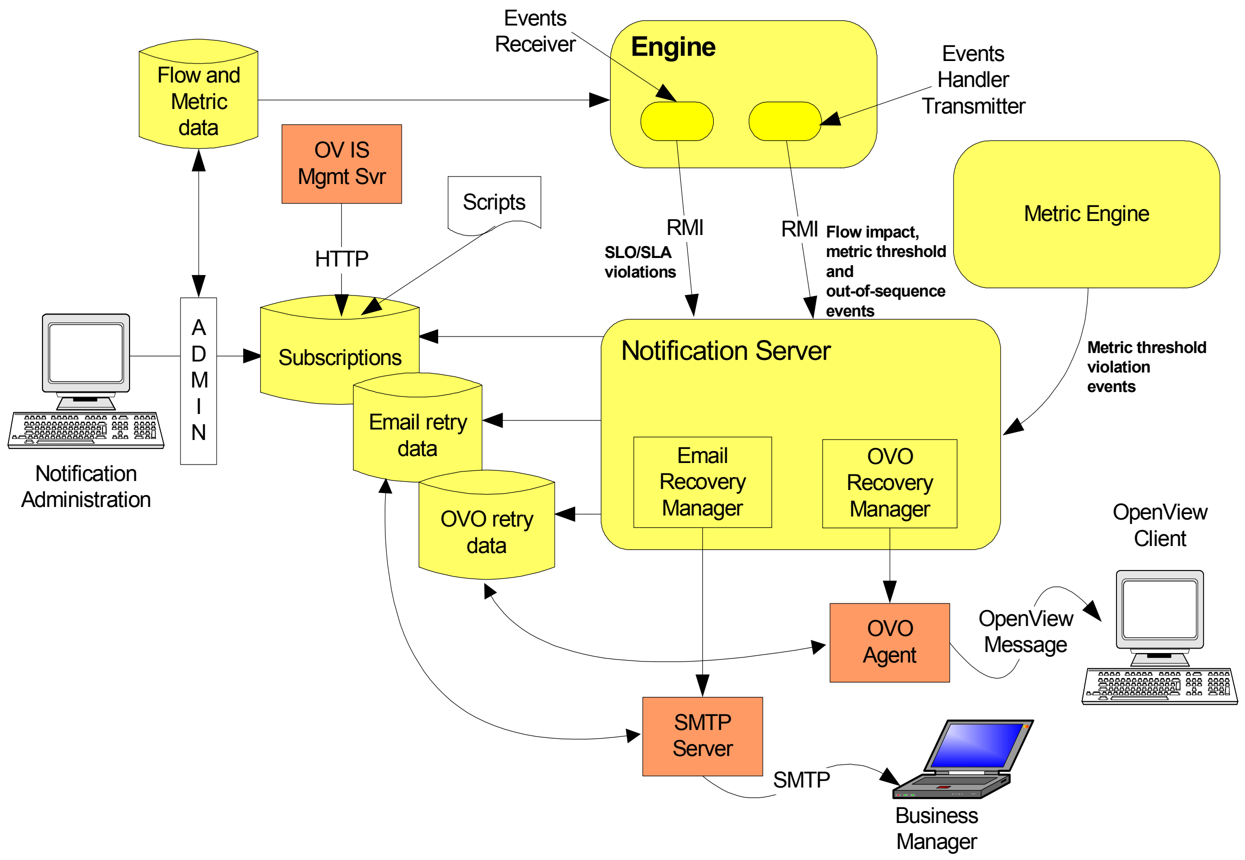
- Configuring scripts that the Notification Server runs when it receives a specific notification event; see section Creating Scripts on page 241.

You can also change the login username and password for the Notification Server Administration Console; see section Appendix B, Servlet Container Authentication.

Follow the instructions in the section appropriate to the task that you want to complete.

The Figure 22 shows an architectural overview of the Notification Server.

Figure 22 Notification Server Architecture



Adding Users for Email Subscriptions

Complete the following steps to configure the Notification Server to send email notifications of impacted flows to a specified email account.

1. Make sure the `Servlet Engine` component is started using the OVBPI Administration Console.
2. Start the OVBPI Notification Server Administration as follows:
   a. Open a new Web browser window
   b. Type the following URL:

      `http://`*`hostname`*`:44080/ovbpinotifyadmin`

      where:

      – *`hostname`* is the fully qualified hostname of the system where the OVBPI Server is running.

      – `44080` is the port number for the Servlet Engine, identified by the `ServletEngine HTTP` port number. Use the port number configured for your system.

      If you are running the Web Browser on the same system as the OVBPI Server, you can use `localhost` in the URL.
   c. You are presented with a dialog to enter the login credentials for the Notification Server Administration Console. Enter a User Name and Password for the Console. By default the User Name is `admin` and the Password is `ovbpi`. Appendix B, Servlet Container Authentication describes how to change the username and password credentials for the Notification Server Administration Console.
3. Select `Email Subscriptions` from the menu.

4. Click the `New User` button to add a new user.

   You are presented with a `User Information` screen.

5. On the `User Information` screen:

   a. Enter a `User Name`

      This is a name that you want to assign to the user; it can be any name that identifies the user subscription.

   b. Enter an `Email Address`

      This is the email address for the user named in the previous step. This must be a valid email account.

   c. Click `OK`.

      The new user now has an account on the Notification Server and is listed in the User List with the other users.

You now have the option to:

- add another new user as described in the above steps.
- subscribe the new user account to the events that you want it to receive. To do this refer to section Configuring the Events Received by Notification Server Users on page 213.
- return to the main menu, where you can add more email subscriptions or add OpenView message subscriptions.
- delete an entry in the User List. To do this select the checkbox next to the `User Name` entry that you want to delete and then click the `Delete` button.
- logout of the Notification Server Administration Console. You do this by closing the Web Browser Window where the console is running.

Configuring the Events Received by Notification Server Users

When you have added the user accounts for the users to receive notifications, you then need to configure these user accounts to subscribe to OVBPI events that you want them to receive. To do this, select the `Subscriptions` link on the Users List screen, under the `Email Subscriptions` column, for the user that you want to configure. This link takes you to the `Email Subscriptions` screen.

You are presented with three options:

- `Flow Subscriptions`; see section "Flow Subscription" on page 213.

  Select this option if you want to add user accounts that you want to configure to subscribe to flow impact, out-of-sequence, or metric threshold alerts.

- `OVIS Service Level Agreement Subscriptions`; see section "OVIS Service Level Agreement Subscription" on page 216.

  This option is available only when you choose to enable OVIS interoperability using the OVBPI Administration Console. Select this option if you want to add accounts to subscribe to SLA alerts.

- `OVIS Service Level Objectives Subscriptions`; see section "OVIS Service Level Objectives Subscription" on page 217.

  This option is available only when you choose to enable OVIS interoperability using the OVBPI Administration Console. Select this option if you want to add accounts to subscribe to SLO alerts.

Flow Subscription

To add a flow impact, out-of-sequence, or metric threshold alert subscription to a user account on the Notification Server, complete the following steps:

1. Select `Flow Subscriptions` from the list of options on the `Email Subscription` screen for the user account that you are modifying.

   You are taken to the `Flow Subscriptions` screen.

2. Click the `New Subscription` button.

   You are taken to the `New Event Subscription` screen where you can add the email subscription details.

3. Select the Event Name and the Flow Name for your new event subscription:

   — Event Name:

     You can select from: `Flow Impacted` events, `Flow Metric Threshold` Events, `Flow Out Of Sequence` events, or `All` events. If you select `All` you subscribe to all types of event.

   — Flow Name:

     You can select all flow names to report on, or you can choose a specific flow that is deployed to the Business Impact Engine. There is a drop down list provided, which lists all the deployed flows for you to select from, including the `All` option.

4. Click `Next` to move to the next screen, where you select the Minimum Severity Level and the Template for your new subscription:

   — Minimum Severity Level:

     This is the minimum severity level for which you want to receive email notifications; for example, if you select Critical you will receive email notifications only for alerts that are critical. If you select Minor, you receive email notifications for Minor, Warning, Major and Critical alerts.

   — Template:

     Select the template that is appropriate to your Event Name, for example, you might have created your own template for the event. If you have selected `Flow Impacted` as the Event Name, choose the `EMAIL-FlowImpactedDefault.vm` template. If you have selected `Flow Metric Threshold` as the Event Name, choose the `EMAIL-MetricThresholdDefault.vm` template. If you have selected `Flow Out of Sequence`, choose the `EMAIL-OutOfSequenceDefault.vm` template. If you have selected `All` for the Event Name, choose the `EMAIL-GenericDefault.vm` template, as it provides the most detailed information.

➤ If you create two subscriptions for the same event type, each with different templates, OVBPI uses the template from the more restrictive subscription to format the alert. For example, if you create a subscription for all flow impact events using one template and a subscription for a specific flow impact event using a different template; the template for the specific flow impact event is the one used to format the alert.

5. Click `OK` to commit the subscription to the administration database.

   The new Event Subscription is now added to the list of Flow Subscriptions for the user.

You now have the option to:

- add another subscription as described above.
- return to the Email Subscription page; click the `Email Subs` button or link.
- delete an entry in the subscription list. To do this select the checkbox next to the entry that you want to delete and then click the `Delete` button.
- logout of the Notification Server Administration Console. You do this by closing the Web Browser Window where the console is running.

OVIS Service Level Agreement Subscription

This option is available when you choose to enable OVIS interoperability.

To add an SLA event subscription to a user account on the Notification Server, complete the following steps:

1. Select `OVIS Service Level Agreement Subscriptions` from the list of options on the `Email Subscriptions` screen for the user account that you are modifying.

   You are taken to the `OVIS SLA Subscription` screen.

2. Click the `New Subscription` button.

   You are taken to the `OVIS SLA Subscriptions Details` screen where you are asked to select the customer for the SLA subscription. The Customer details are those defined in the OVIS database.

   ➤ The Notification Server presents only those OVIS Customers that have Service Level Agreements defined. You can use the `Reload OVIS Config` option to update the list.

3. Select one of the defined customers, or select `All` if you want to filter for all SLA violations defined in OVIS.
4. Click OK to move to the next screen, where you are asked to select a service level agreement.
5. Select an SLA by name, or select `All` if you want the user to receive alerts for all SLA violations.
6. Click `OK` to move the next screen where you can select a template for the email message that contains the details of the SLA violation.
7. Select a template that is appropriate to the SLA subscription, for example, the default SLA template (`EMAIL-OVIS-SLA-Default.vm`), or select a custom template, if you have created one.
8. Click `OK` to complete the subscription and move to the next screen that lists the SLA subscriptions for the user account that you are modifying.

You now have the option to:

- add another OVIS SLA Subscription as described above.
- return to the Email Subscriptions page, in which case, click the `Email Subs` button, or link.
- delete an entry in the OVIS SLA Subscription list. To do this select the checkbox next the entry that you want to delete and then click the `Delete` button.
- logout of the Notification Server Administration Console. You do this by closing the Web Browser Window where the console is running.

OVIS Service Level Objectives Subscription

To add an SLO subscription to a user account on the Notification Server, complete the following steps:

1. Select `OVIS Service Level Objective Subscriptions` from the list of options on the `Email Subscriptions` screen for the user account that you are modifying.

   You are taken to the `OVIS SLO Subscriptions` screen.

2. Click the `New Subscription` button.

   You are taken to the `OVIS SLO Subscription Details` screen where you are asked to select the customer for the SLO subscription. The Customer details are those defined in the OVIS database.

3. Select one of the defined customers, or select `All` if you want to filter for all SLO violations defined in OVIS.
4. Click `OK` to move to the next screen, where you are asked to select a service group.
5. Select the Service Group name from the drop-down list provided, or select `All` if you want to filter on all OVIS Service Groups.
6. Click `OK` to move to the next screen, where you are asked to select an objective metric for the SLO.
7. Select an objective metric name from the drop-down list, or select `All` to filter on all objective metrics.
8. Click `OK` to move to the next screen.

9. Select a template that is appropriate to SLO subscription, for example, the default SLO template (`EMAIL-OVIS-SLO-Default.vm`), or select a custom template, if you have created one.
10. Click `OK` to move to the next screen that lists the SLO subscriptions for the user.

You now have the option to:

- add another OVIS SLO Subscription as described above.
- return to the Email Subscription page; click the `Email Subs` button or link.
- delete an entry in the OVIS SLO Subscription list. To do this select the checkbox next the entry that you want to delete and then click the `Delete` button.
- logout of the Notification Server Administration Console. You do this by closing the Web Browser Window where the console is running.

Adding OpenView Operations Message Subscriptions

The Notification Server uses Velocity and XSLT templates to create OpenView Operations messages before forwarding the message to the OpenView Operations agent.

The OpenView Operations agent then applies the OpenView templates to filter out unwanted messages and to make any additional changes to the OpenView Operations message format.

Before configuring OVBPI to send events to OpenView, ensure that the OpenView Operations Agent and Message Interface template are installed and set up on the system where OVBPI and the Notification Server are installed. Refer to the *OpenView VantagePoint Operations for Unix Administrator's Reference Guide*.

Complete the following steps to configure the Notification Server to send OpenView Operations message notifications to the OpenView Agent for flow impact, out-of-sequence or metric threshold notification events:

1. Make sure the `Servlet Engine` component is started.
2. Start the OVBPI Notification Server Administration as follows:
   a. Open a new Web browser window
   b. Type the following URL:

      `http://`*`hostname`*`:44080/ovbpinotifyadmin/index.jsp`

      – where *`hostname`* is the fully qualified hostname of the system where the OVBPI Server is running.

      – `44080` is the port number for the Servlet Engine, identified by the `ServletEngine HTTP` port number. Use the port number configured for your system.

      If you are running the Web Browser on the same system as the OVBPI Server, you can use `localhost` in the URL.
   c. You are presented with a dialog to enter the login credentials for the Notification Server Administration Console. Enter a User Name and Password for the Console. By default the User Name is `admin` and the

Password is `ovbpi`. Appendix B, Servlet Container Authentication describes how to change the username and password credentials for the Notification Server Administration Console.

3. Select `OVO Message Subscriptions` from the menu.

   You are taken to the `OVO Subscriptions` screen.

4. Click the `New Subscription` button.

   You are presented with the `New Event Subscription` Screen where you can enter the subscription details for the OpenView operations messages.

5. From the `New Event Subscription` screen, select the Event Name and the Flow Name for your new event subscription:

   — Event Name

   You can select to filter for `Flow Impacted` events, `Flow Metric Threshold` events, `Flow Out Of Sequence` events, or `All` events. If you select `All`, you receive OpenView impact alerts for all OVBPI notification events.

   — Flow Name

   You can select all flow names to report on, or you can choose a specific flow that you have defined. There is a drop down list provided, which lists all the deployed flows for you to select from, including the `All` option.

6. Click Next to move to the next screen, where you select the Minimum Severity Level and the Template for your new subscription:

   — Minimum Severity Level

   This is the minimum severity level for which you want to receive email notifications; for example, if you select Critical you will receive email notifications only for alerts that are critical. If you select Minor, you receive email notifications for Minor, Warning, Major and Critical alerts.

   — Template

   Select the template that is appropriate to your Event Name, for example you might have created a custom template. If you have selected `Flow Impacted` as the Event Name, choose the `OVO-FlowImpactDefault.vm` template. If you have selected `Flow Metric Threshold` as the Event Name, choose the `OVO-MetricThresholdDefault.vm` template. If you have selected `Flow`

`Out of Sequence`, choose the template for the `OVO-OutOfSequence.vm` for the template. If you have selected `All` for the Event Name, choose the `OVO-GenericDefault.vm` template, as it provides the most detailed information.

If you create two subscriptions for the same OVO message subscription, each with different templates, OVBPI uses the template from the more restrictive subscription to format the alert. For example, if you create a subscription for all flow impact events using one template and a subscription for a specific flow impact event using a different template; the template for the specific flow impact event is the one used to format the alert.

7. Click `OK` to commit the subscription to the Notification Server Administration database.

   The new OVO Message Subscription is now added to the list of OVO Subscriptions.

You now have the option to:

- add another OVO Message Subscription as described above.
- return to the Main Menu; click the `Menu` button, or link.
- delete an entry in the OVO Message Subscription list. To do this select the checkbox next the entry that you want to delete and then click the `Delete` button.
- logout of the Notification Server Administration Console. You do this by closing the Web Browser Window where the console is running.

You can configure the template set used by OpenView and you can filter the events received through OpenView by customizing the OpenView Operations Message Interface template. Details of how to do this are provided in the OpenView documentation.

Script Subscription

You can configure the Notification Server to run a script when it receives specific notification events. Section "Creating Scripts" on page 241 describes how you can create these scripts and provides details of the variables that you can include in your scripts to report details of the alert as required.

This section describes how to configure the Notification Server to run the scripts for specific notification alerts that it processes for flow impact, out-of-sequence or metric threshold notification events:

1. Make sure the `Servlet Engine` component is started.
2. Start the OVBPI Notification Server Administration as follows:
   a. Open a new Web browser window
   b. Type the following URL:

      `http://`*`hostname`*`:44080/ovbpinotifyadmin/index.jsp`

      where:

      – *`hostname`* is the fully qualified hostname of the system where the OVBPI Server is running.

      – `44080` is the port number for the Servlet Engine, identified by the `ServletEngine HTTP` port number. Use the port number configured for your system.

      If you are running the Web Browser on the same system as the OVBPI Server, you can use `localhost` in the URL.
   c. You are presented with a dialog to enter the login credentials for the Notification Server Administration Console. Enter a User Name and Password for the Console. By default the User Name is `admin` and the Password is `ovbpi`. Appendix B, Servlet Container Authentication describes how to change the username and password credentials for the Notification Server Administration Console.
3. Select `Script Subscriptions` from the menu.

   You are taken to the `Script Subscriptions` screen.
4. Click the `New Subscription` button.

   You are presented with a `New Script Subscription` screen where you can enter the subscription details for the script.

5. From the `New Script Subscription` screen, enter the following details:

   — Event Name

   You can select to filter for `Flow Impacted` events, `Flow Metric Threshold` events, `Flow Out Of Sequence` events, or `All` events. If you select `All`, you receive OpenView impact alerts for all OVBPI notification events.

   — Flow Name

   You can select all flow names to report on, or you can choose a specific flow that you have defined. There is a drop down list provided, which lists all the deployed flows for you to select from, including the `All` option.

6. Click Next to move to the next screen where you select the `Minimum Severity Level` and the `Script Name`:

   — Minimum Severity Level

   This is the minimum severity level for which you want to receive email notifications; for example, if you select Critical you will receive email notifications only for alerts that are critical. If you select Minor, you receive email notifications for Minor, Warning, Major and Critical alerts.

   — Script Name

   Select the script that you want the Notification Server to run when it received a notification event for the specified event and flow. All the scripts that you have defined and saved in the `notify-scripts` are presented in the drop-down list for you to select from.

7. Click OK to commit the subscription to the administration database.

   The new Script Subscription is added to the list of Script Message Subscriptions.

You now have the option to:

- add another New Script Subscription as described above.
- return to the Main Menu; click the `Menu` button, or link.

- delete an entry in the Script Subscription list. To do this select the checkbox next the entry that you want to delete and then click the `Delete` button.
- logout of the Notification Server Administration Console. You do this by closing the Web Browser Window where the console is running.

Creating Notification Server Templates

The Notification Server uses templates to format email messages and messages sent to OpenView Operations. This template contains instructions on how to format event information, for example, event name and event time. You might want customize an email template if you want to generate email messages for different locales.

This section describes how OVBPI uses these templates and some guidance in how you can modify them. You need to be familiar with either XSLT, or Velocity to make the changes; XSLT is a standard for transforming XML documents. It is up to you which format you choose.

The Notification Server uses the Apache Velocity template engine to process Velocity templates. There is a user guide available with the template engine that provides details of creating and using Velocity templates. Details of Velocity templates, including a user guide for creating them, can be found at the following location:

`http://jakarta.apache.org/velocity/`

You configure the template used for each event when you create the event subscription through the Notification Server Web Administration Console. The Notification Server then uses the template that you have defined to format all notifications related to the specified event.

Creating Email Templates

You create a custom email template for a specific event. These custom email templates must have a file name that starts with the string `EMAIL-` and a file extension of `.vm` for Velocity templates and `.xsl` for XSLT templates.

The Notification Server reads the custom templates from the following directories:

- *OVBPI-install-dir*`/data/conf/bia/notify-templates/flows`
- *OVBPI-install-dir*`/data/conf/bia/notify-templates/slos`
- *OVBPI-install-dir*`/data/conf/bia/notify-templates/slas`

If you want to modify these templates, make a copy of the appropriate example template from the files in the following directory:

OVBPI-install-dir`/examples/bia/NotifSvrTemplates`

Make sure that you add your template to the correct directory as the Notification Server Administration Console presents the templates in the context of the selection, based on the contents of the above directories. For example, when creating an SLA subscription, you are offered the list of templates from the following directory:

`OVBPI-install-dir/data/conf/bia/notify-templates/slas`

You also need to make sure that you copy the template from the `examples` directory to the `data` directory as the `examples` directory is updated and files replaced if you reinstall OVBPI.

The Notification Server uses an email template to create XML that describes an email. Then the notification server uses the XML to create an email.

The email XML is very simple; it contains a subject, content type, and body. The actual schema is described in the file `email.xsd`, which is located at:

`OVBPI-install-dir/misc/bia`

The Notification Server applies the Velocity or XSLT templates that you define to an alert from the Business Impact Engine. From this the Notification Server is able to create an email XML document based on your templates.

OpenView Operations Templates

The Notification Server uses a template to create XML that describes the OpenView Operations message. It then uses this XML to create an OpenView Operations message.

You configure the templates that are used to send OpenView messages through the Notification Server Administration Console.

Custom OpenView Operations template files must start with the string `OVO-` and a file extension of `.vm` for Velocity templates and `.xsl` for XSLT templates. Store custom templates in the following directory:

`OVBPI-install-dir/data/conf/bia/notify-templates/flows`

The following is an example of an OpenView Operations message XML document:

```
<?xml version="1.0" encoding="UTF-8"?>
<OVOMessage xmlns:xsi="http://www.w3.org/2001/XMLSchema
   instance"xsi:noNamespaceSchemaLocation="ovomessage.xsd" >
<Severity>critical</Severity>
<Application>OVBPI application</Application>
<Object>FLOW_IMPACT or FLOW_OUT_OF_SEQUENCE</Object>
<MessageGroup>NOTIFICATIONS</MessageGroup>
<MessageText>A FLOW_OUT_OF_SEQUENCE alert occurred at 7.10pm.</
MessageText>
<Option>example_option_variable_1=this is an example option
value</Option>
<Option>example_option_variable_2=this is another example option
value</Option>
</OVOMessage>
```

The example OpenView operations message XML document contains an element for each message attribute:

- message severity
- application
- object
- message group
- message text

Optionally, the XML document, can contain message options. An option has the form *variable=value*. Refer to the *OpenView VantagePoint Operations for Unix Administrator's Reference Volume I* for more information about OpenView Operations messages. The notification server applies Velocity and XSLT templates to an alert from the Business Impact Engine to create an OpenView Operations message XML document.

The schema for the message is available at the following location:

`*ovbpi-install-dir*/misc/bia/ovomessage.xsd`

Velocity Templates

This section describes the list of the OVBPI alert methods that have been defined for use within your Velocity template. An alert variable in the Velocity context contains information from an Business Impact Engine event.

A Velocity template is a file that contains the XML describing an email or OpenView Operations message.

In addition to text, the file can contain Velocity formatting directives. The following is an example of an email Velocity template:

```
<?xml version="1.0" encoding="UTF-8"?>
<Email xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
    xsi:noNamespaceSchemaLocation='email.xsd'>
<Subject>
$alert.getSeverity() $alert.getEventName() Alert
</Subject>
<ContentType>text/plain</ContentType>
<Body>
A "$alert.getEventName()" alert has occurred.

Severity: $alert.getSeverity()
Event Group: $alert.getEventGroup()
Event Name: $alert.getEventName()
Event Time: $alert.getEventTime()
Flow Name: $alert.getFlowName()
Flow Guid: $alert.getFlowGuid()
</Body>
</Email
```

A formatting directive starts with a hash (#) or a dollar ($); these formatting directives are described more fully in the *Velocity Users' Guide*. A formatting directive is used to insert event information into the XML document. In the case of OVBPI this event information is provided through the alert variables.

The `$alert` variable contains information from a Business Impact Engine event, for example, the service name for the event. The `$alert` variable contains methods such as `getFlowName()` and `getServiceName()`, which can be used to insert the event flow name and the event service name into the XML. (`getServiceName()` inserts the service name).

➤ If the Velocity templates that you create contain non-ASCII characters, you must encode and save the template as UTF8. This UTF8 needs to be created without a byte order mark (BOM). BOMs are added automatically by some Windows editors; for example, Notepad and Wordpad. Choose an editor that enables you to exclude the BOM.

If you include a BOM in your template encoding, it is not recognized as an XML file.

When the Notification Server generates an email notification message, it adds data to the message, where the display language is determined by the locale of the OVBPI Server. If your email recipients are not in the same locale as the OVBPI Server, you can use the Velocity templates to specify the locale for the language used for the email notifications. This enables you to provide email notifications in several languages if required.

The java Locale() class is used to identify the language string used within the alert methods. The definition for the Locale() class can be found at the following URL:

http://java.sun.com/j2se/1.4.2/docs/api/java/util/Locale.html

The Locale() class takes a parameter comprising the language code and the country code as described in the definition for the Locale() class; for example: ko_KR for Korea and en_CA English-speaking Canadian.

Methods for All Alerts

The following table lists the alert methods that can be used for all alerts: Flow Impact, Flow Out-Of-Sequence, Business Metric, Metric Threshold, SLO and SLA.

Table 39 Velocity Template Alert Methods for all Alerts

Alert Method Name	Alert Method Description
$alert.getVersion()	Returns a STRING containing the version number for the OVBPI event.
$alert.getSeverity(*locale*)	Returns a STRING containing the severity of the OVBPI event. Possible values are `Critical`, `Major`, `Minor`, `Warning`, and `Normal`. The language used for these values is determined by the locale for the OVBPI Server or, if specified, the locale of the country and language specified in *`locale`*, for example: • $alert.getSeverity("ja"), for a Japanese locale • $alert.getSeverity("ko"), for a Korean locale The value returned is also used to determine the color of the text in the email notification.
$alert.getEventGroup()	Returns a STRING containing the name of the event group for the OVBPI event, that is, `NOTIFICATIONS`.
$alert.getEventName()	Returns a STRING containing the name of the OVBPI event, that is, `FLOW_IMPACT`, `FLOW_METRIC_THRESHOLD_ALERT` or `FLOW_OUT_OF_SEQUENCE`. In the case of an SLO, this is the objective identifier. In the case of an SLA, this is the SLA identifier.
$alert.getEventTime(*locale*)	Returns a STRING containing the time of the OVBPI event, formatted using the locale for the OVBPI Server or, if specified, the locale of the country and language specified in *`locale`*.
$alert.getEventDataListSize()	Returns the number of event data items in the OVBPI event.

Table 39 Velocity Template Alert Methods for all Alerts

Alert Method Name	Alert Method Description
$alert.getEventDataNames()	Returns an ARRAYLIST of the names of the event data items in the OVBPI event.
$alert.getEventDataByName (*Stringname, locale*)	Returns a STRING containing the value of the data item with the given name using the locale for the OVBPI Server or, if specified, the locale of the country and language specified in *locale*.
$alert.getEventDataByIndex (*index, locale*)	Returns a STRING containing the value of the data item located at the supplied index using the locale for the OVBPI Server or, if specified, the locale of the country and language specified in *locale*.

Methods Specific to Flow Impact Alerts

The following table lists the alert methods that can be used for Flow Impact alerts. Using any of these methods for other alerts results in a message containing erroneous data.

Table 40 Velocity Template Alert Methods for Flow Impact Alerts Only

Alert Method Name	Alert Method Description
$alert.getFlowName()	Returns a STRING containing the flow name for the OVBPI event.
$alert.getFlowGuid()	Returns a STRING containing GUID for the flow definition in the OVBPI event.
$alert.getServiceName()	Returns a STRING containing the name of the service for the OVBPI event.
$alert.getServiceGuid()	Returns a STRING containing the service GUID for the OVBPI event.

Table 40 Velocity Template Alert Methods for Flow Impact Alerts Only

Alert Method Name	Alert Method Description
$alert.getFlowNodeString()	Returns a STRING containing node names for the OVBPI event. An example of a string returned by this method is: `Bill Customer`, `Check Stock`, `Remove Item`.
$alert.getFlowNodeList()	Returns an ARRAYLIST of node names in the OVBPI event. You can use the methods defined in the ArrayList class. For example, you can access the first element of an array list named flowNodeList using `$flowNodeList.get(0)`
$alert.getFlowNodeListSize()	Returns the number of elements in the flow node list in the OVBPI event.
$alert.getRootCause()	Returns a STRING containing the labels of the root cause services in the OVBPI event. (Note that this string cannot be localized.) An example of a string returned by this method is: `Oracle Financial System-&gt;Oracle System Europe-&gt;Network Resources`

Table 40 Velocity Template Alert Methods for Flow Impact Alerts Only

Alert Method Name	Alert Method Description
$alert.getFlowNodeStatusList()	Returns an ARRAYLIST of the status of the Nodes in the OVBPI event. You can use the methods defined in the ArrayList class. For example, you can access the first element of an array list namesFlowNodeStatusList using `$flowNodeStatusList.get(0)`.
$alert.getFlowNodeStatusString (*locale*)	Returns a STRING containing the labels for the node status in the OVBPI event. For example, `Critical`, `Major`, `Minor`, `Warning` and `Normal`. These status correspond to the OpenView Operations severity levels. The language used for these values is determined by the locale for the OVBPI Server or, if specified, the locale of the country and language specified in *`locale`*, for example: • $alert.getFlowNodeStatusString("ja"), for a Japanese locale • $alert.getFlowNodeStatusString("ko"), for a Korean locale
$alert.getFlowNodeStatusList Size()	Returns the number of elements in the flow node status list in the OVBPI event.

Methods Specific to Out-of-Sequence Alerts

The following table lists the methods for out-of-sequence events only. Using any of these methods for other alerts results in a message containing erroneous data.

Table 41 Velocity Template Methods for Out-of-Sequence Alerts Only

Alert Method Name	Alert Method Description
$alert.getFlowName()	Returns a STRING containing the flow name for the OVBPI event.
$alert.getFlowGuid()	Returns a STRING containing GUID for the flow definition in the OVBPI event.
$alert.getFlowInstanceGuid()	Returns a STRING containing GUID for the flow instance in the OVBPI event.
$alert.getSourceNode()	Returns a STRING containing the name of the node that is the source node for the out-of sequence alert.
$alert.getDestinationNode()	Returns a STRING containing the name of the node that is the destination node for the out-of sequence alert.
$alert.getFlowIdentifier()	Returns a STRING containing the identifier for the impacted flow instance.

Methods Specific to Metric and Threshold Alerts

The following table lists the metric and threshold alert notification events only. Using any of these methods for other alerts results in a message containing erroneous data.

Table 42 Velocity Template Methods for Metric and Threshold Alerts Only

Alert Method Name	Alert Method Description
$alert.getThresholdAlertGuid()	Returns a STRING containing the internal unique identifier (GUID) of the threshold alert event raised for a specific threshold violation.
$alert.getThresholdGuid()	Returns a STRING containing the internal unique identifier (GUID) for the metric threshold defined for the business process metric in the OVBPI event.
$alert.getThresholdName()	Returns a STRING containing metric threshold name in the OVBPI event.
$alert.getThresholdMessage()	Returns a STRING containing the metric threshold message in the OVBPI event. This is the message that you have defined to be displayed when the event is triggered.
$alert.getFlowMetricGuid()	Returns a STRING containing the business process metric GUID in the OVBPI event.
$alert.getFlowMetricValue()	Returns a STRING containing the business process metric value in the OVBPI event.
$alert.getFlowMetricName()	Returns a STRING containing the business process metric name in the OVBPI event.

Methods Specific to OVIS SLA Violation Methods

The following table lists the OVIS SLA Violation methods that can be used for OVBPI. Using any of these methods for other alerts results in a message containing erroneous data.

Table 43 Velocity Template Methods for SLO Violations

SLO Method Name	SLO Method Description
$alert.getOvisCustomer()	Returns a STRING containing the name of the OVIS Customer defined in the OVIS Configuration Manager that triggered the violation.
$alert.getOvisServiceGroup()	Returns a STRING containing the name of the OVIS Service Group defined in the OVIS Configuration Manager that triggered the violation.
$alert.getOvisHost()	Returns a STRING containing the name of the host that OVIS is monitoring and that triggered the violation; for example, the name of a Web site.
$alert.getOvisProbe()	Returns a STRING containing the location of the probe that relates to the violation.
$alert.getOvisViolationTime (*locale*)	Returns a STRING containing the date and time that the violation was reported within OVIS. The date and time are displayed in the locale of the OVBPI Server or, if specified, the locale of the country and language specified in *locale*
$alert.getOvisObjectiveName()	Returns a STRING containing the name of the SLO metric relating to the violation.
$alert.getOvisObjectiveId()	Returns a STRING containing the unique identifier for the SLO metric relating to the violation.
$alert.getOvisObjective Operation()	Returns a STRING containing the arithmetical operation that is defined for the SLO that has been violated.

Table 43 Velocity Template Methods for SLO Violations

SLO Method Name	SLO Method Description
$alert.getOvisObjectiveValue()	Returns a DOUBLE that is the metric value recorded for the SLO that has been violated.
$alert.getOvisObjective Threshold()	Returns a DOUBLE that is the metric threshold value assigned to the SLO that has been violated.

Methods Specific to OVIS SLA Violations

The following table lists the OVIS SLA Violation methods that can be used for OVBPI. Using any of these methods for other alerts results in a message containing erroneous data.

Table 44 Velocity Template Methods for SLA Violations

SLA Method Name	SLA Method Description
$alert.getOvisAgreementName()	Returns a STRING containing the name of the SLA that has been violated.
$alert.getOvisAgreementID()	Returns a STRING containing the unique identifier for the SLA that has been violated.
$alert.getOvisCustomer()	Returns a STRING containing the Customer details relating to the SLA that has been violated.
$alert.getOvisViolationTime()	Returns a STRING containing the time that the SLA was violated.
$alert.getOvisAgreement ConformanceThreshold()	Returns a DOUBLE that contains the conformance threshold for the SLA, expressed as a percentage.
$alert.getOvisAgreement Conformance()	Returns a DOUBLE that contains the conformance for the SLA, expressed as a percentage.

Table 44 Velocity Template Methods for SLA Violations

SLA Method Name	SLA Method Description
$alert.getOvisAgreement Samples()	Returns an INTEGER that contains the number of samples taken over the measurement period for the violation.
$alert.getOvisAgreement ConformingSamples()	Returns an INTEGER that contains the number of samples that conformed to the SLA over the measurement period.

XSLT Templates

An XSLT template is a file containing XML that transforms event XML into email or OVO message XML. An example of an email XSLT template is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0"
        xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:etype="http://www.hp.com/openview/bia/eventtype"
        xmlns:edata="http://www.hp.com/openview/bia/eventdata">
  <xsl:output method="xml"
       indent="no" />

<xsl:template match="/">
<Email xsi:noNamespaceSchemaLocation='email.xsd'>
<Subject>
<xsl:apply-templates select="//env:Body/*/etype:Severity/*"/><xsl:text> </
xsl:text><xsl:value-of select="//etype:EventName" /> Alert</Subject>
<ContentType>text/plain</ContentType>
<Body>
A "<xsl:value-of select="//etype:EventName" />" alert has occurred.

Severity: <xsl:apply-templates select="//env:Body/*/etype:Severity/*"/>
Event Group: <xsl:value-of select="//etype:EventGroup" />
Event Name: <xsl:value-of select="//etype:EventName" />
Event Time: <xsl:value-of select="//etype:EventTime" /><xsl:text>
</xsl:text>
<xsl:for-each select="//edata:DataItem" >
<xsl:choose>
<xsl:when test="edata:Name = 'RootCause'" >
<xsl:text>Root Cause: </xsl:text>
```

```
<xsl:for-each select="//edata:Value/Services/Service" >
<xsl:text>-&gt;</xsl:text><xsl:value-of select="Label" />
</xsl:for-each>
</xsl:when>
<xsl:otherwise>
<xsl:value-of select="edata:Name" /><xsl:text>: </xsl:text><xsl:value-of
select="edata:Value" />
</xsl:otherwise>
</xsl:choose>
<xsl:text>
</xsl:text>
</xsl:for-each>
</Body>
</Email>
</xsl:template>

<xsl:template match="etype:Critical" >
<xsl:text>Critical</xsl:text>
</xsl:template>

<xsl:template match="etype:Major" >
<xsl:text>Major</xsl:text>
</xsl:template>

<xsl:template match="etype:Minor" >
<xsl:text>Minor</xsl:text>
</xsl:template>

<xsl:template match="etype:Warning" >
<xsl:text>Warning</xsl:text>
</xsl:template>

<xsl:template match="etype:Normal" >
<xsl:text>Normal</xsl:text>
</xsl:template>

</xsl:stylesheet>
```

The style sheet is applied to the event sent by the Business Impact Engine. The style sheet contains email XML text as well as XSL elements. XSL elements format information from the form of the event and insert it into the email XML.

The schemas for the event are located in the files `eventtype.xsd` and `eventdata.xsd` in the following directory:

`OVBPI-install-dir``/misc/bia`

The message header contains a business event type, which is described in `eventtype.xsd`. The message body contains a business event type followed by business event data; Business event data is described in `eventdata.xsd`.

Creating Scripts

You can define a script, which is a Windows `.bat` file, to be run when a notification event is received by the Notification Server; for example, this script might send an SMS message to a recipient directly (bypassing the email server), or it might update an entry in a database or spreadsheet.

You can create a script to complete any action that you want, and you can also include any of the environment variables that have been defined for use within your scripts. These environment variables identify configuration information relating to the OVBPI data available for notification alerts and are defined in section "Environment Variables for Scripts" on page 244.

These environment variables are included in your scripts using the following notation:

`%env_name%`

where `%env_name%` is environment variable name, for example, `%OVBPI_FLOWNAME%`.

The following are two examples of how to create a script and configure the Notification Server to run the script when a specific notification event is received.

Example Script to Write a String to a File

The following is an example of the steps that you need to complete in order to configure the Notification Server to write the string `Hello World` to a file on receipt of a specific flow impact event:

1. Create the script that you want to be executed when the notification event is received, for example:

   ```
   echo “hello world” >>c:\notif.txt
   ```

   This script writes the string “`hello world`” to the file `notif.txt` in the root of your `c:` drive.

2. Save the script to the following location with a unique file name and a `.bat` file extension:

   ```
   OVBPI-install-dir/data/conf/bia/notify-scripts
   ```

   The Notification Server looks in this directory for scripts and presents these scripts to your through the Web Administration Console for the Notification Server when you are configuring subscriptions. Note that if the script that you create does not have a `.bat` file extension, it is ignored.

3. Follow the instructions in section “Script Subscription” on page 222 to create a subscription for the script selecting `Flow Impacted` as the `Event Name` and the name of the flow for the `Flow Name`.

   If you have the Web Administration for the Notification Server administration open at the New Event Subscription page, you might need to click the Refresh option on your Web Browser. This refreshes the page and adds the new script to the list.

Example Script to Write Flow Name and Blocked Instances to a File

The following is an example of the steps that you need to complete in order to configure the Notification Server to take the flow name and the number of blocked instances from the event data and write the information to a file on receipt of a specific flow impact event:

1. Create the script that you want to be executed when the notification event is received, for example:

```
echo %OVBPI_FLOWNAME%, %OVBPI_NOOFBLOCKEDINSTANCES%>>c:\notif2.txt
```

   This script writes the values of the flow name and the number of blocked instances to the file `notif2.txt` in the root of your `c:` drive.

2. Save the script to the following location with a unique filename:

   `OVBPI-install-dir/data/conf/bia/notify-scripts`

   The Notification Server looks in this directory for scripts and presents these scripts to your through the Web Administration Console for the Notification Server when you are configuring subscriptions.

3. Follow the instructions in section "Script Subscription" on page 222 to create a subscription for the script selecting `Flow Impacted` as the `Event Name` and the name of the flow for the `Flow Name`.

   If you have the Web Administration for the Notification Server administration open at the New Event Subscription page, you might need to click the Refresh option on your Web Browser. This refreshes the page and adds the new script to the list.

Environment Variables for Scripts

The following tables list the environment variables that can be used within your scripts to report data on flow impact, flow out-of-sequence and metric threshold alerts.

Table 45 on page 244 lists the environment variables that are available for all OVBPI alerts.

Table 45 Environment Variables for All Alerts

Variable	Description
`OVBPI_EVENTGROUP`	name of the event group, which is always `NOTIFICATIONS`.
`OVBPI_EVENTNAME`	name of the event: `FLOW_IMPACT`, `FLOW_METRIC_THRESHOLD_ALERT` or `FLOW_OUT_OF_SEQUENCE`.
`OVBPI_SEVERITY`	severity of the OVBPI event. Possible values are `Normal`, `Warning`, `Minor`, `Major` or `Critical`

Table 46 on page 245 lists the environment variables available for flow impact alerts. Using these environment variables for other alerts results in erroneous data being returned.

Table 46 Environment Variables for Flow Impact Alerts

Variable	Description
`OVBPI_FLOWGUID`	internal unique identifier (GUID) for the flow definition in the OVBPI event. This identifier is useful to add to a URL in order to link directly to a flow definition page within the Business Process Dashboard.
`OVBPI_FLOWNAME`	name of the flow in the OVBPI event
`OVBPI_NOOFBLOCKEDINSTANCES`	number of `blocked` flow instances at the time of the `FLOW_IMPACT` event. A blocked instance is an instance of the flow that is active at a node and cannot proceed as there is a problem in an underlying operational service.
`OVBPI_NOOFIMPEDEDINSTANCES`	number of `at risk` flow instances at the time of the `FLOW_IMPACT` event. An `at risk` instance is where one or more flow instance has the potential to be blocked in the future.
`OVBPI_SERVICEGUID`	internal unique identifier (GUID) for the service in the OVBPI `FLOW_IMPACT` event.
`OVBPI_SERVICENAME`	name for the service in the OVBPI `FLOW_IMPACT` event.
`OVBPI_ROOTCAUSE`	the label of the root cause service, or services, in the OVBPI `FLOW_IMPACT` event.

Table 47 on page 246 lists the environment variables available for flow metric and threshold alerts. Using these environment variables for other alerts results in erroneous data being returned.

Table 47 Environment Variables for Flow Metric and Threshold Alerts

Variable	Description
OVBPI_FLOWMETRICGUID	internal unique identifier (GUID) for the flow metric in the OVBPI FLOW_METRIC_THRESHOLD_ALERT event.
OVBPI_FLOWMETRICNAME	name of the flow metric in the OVBPI FLOW_METRIC_THRESHOLD_ALERT event.
OVBPI_FLOWMETRICVALUE	value of the flow metric in the OVBPI FLOW_METRIC_THRESHOLD_ALERT event.
OVBPI_THRESHOLDALERTGUID	internal unique identifier of the threshold identified in the OVBPI FLOW_METRIC_THRESHOLD_ALERT event.
OVBPI_THRESHOLDMESSAGE	message defined for the OVBPI FLOW_METRIC_THRESHOLD_ALERT event. This is the message that you have defined to be displayed when the event is triggered.
OVBPI_THRESHOLDNAME	name of the metric threshold in the OVBPI FLOW_METRIC_THRESHOLD_ALERT event.

Table 48 on page 247 lists the environment variables for the Out-of-Sequence alerts. Using these environment variables for other alerts results in erroneous data being returned.

Table 48 Environment Variables for Out-of-Sequence Alerts

Variable	Description
`OVBPI_FLOWIDENTIFIER`	identifier for the flow instance in the OVBPI event. This the property that you have nominated to be the identifier for the flow.
`OVBPI_FLOWINSTANCEGUID`	internal unique identifier (GUID) for the flow instance in the OVBPI event.
`OVBPI_SOURCENODE`	name of the node that is the source node for the out-of-sequence alert.
`OVBPI_DESTINATIONNODE`	name of the node that is the destination node for the out-of-sequence alert.

10

Intervention

This chapter describes how you can make changes to the business flow data, and delete completed instances of Flows and Data definitions using OVBPI tools and configuration options. Specifically this chapter describes:

- modifying, or removing individual instances, services or OVBPI Metrics; see section Intervention Client on page 251
- regularly pruning completed entries from the database; see section Engine Flow and Data Instance Cleaner Parameters on page 257.

Tools and Parameter Settings

There are two different ways of clearing OVBPI data from the database and you need to select the method that is most appropriate to your requirements.

You can use:

- the Intervention Client to remove individual flow instances or data instances from the database.

  The purpose of the Intervention Client is to enable you to access deployed Flow and Data definitions, plus Flow instances, Data instances and Services. This then enables you to modify or delete (with the exception of Services) instances of these definitions in order to correct:

  — potential errors within your OVBPI system.

  — errors in the system that is being monitored.

  — errors that exist within deployed Flow definitions.

  You can delete individual instances using the Intervention Client. However, if you want to delete numbers of completed or active instances regularly you can use the Engine Instance Cleaner parameters as described in section Engine Flow and Data Instance Cleaner Parameters on page 257.

  ➤ You cannot delete services using the Intervention Client; however, you can modify the status of service definitions using the Intervention Client.

- the Engine Instance Cleaner parameters for regular removal of completed instances from the database.

  The instance cleaner parameters are not designed to enable individual instances to be manipulated nor do they enable you to name the flow definitions or data definitions whose instances are to be deleted; these functions are provided by the Intervention Client.

Intervention Client

The Intervention Client enables you to access, or intervene in, business flow instances; this, in turn, enables you to make changes to the flow instances. The Intervention Client also enables you to delete Flow and Data definitions, and clear the average and total counts associated with a specific definition.

The following are some examples of the uses of the Intervention Client:

- where a flow instance has progressed through a flow, but one or more nodes that should be complete still appear as active.

  You can use the Intervention Client to change the state of the active nodes to `Complete`. Nodes can get into this state if you have manually progressed a Flow instance, for example, when an application that usually provides the service is not available and you have substituted another application to complete the step.

- when the start and complete conditions for a node are incorrect and therefore cannot be satisfied. In this case, you might want to delete all instances of the flow and any associated flow data.
- where you want to progress a specific flow instance to its completion, for example, there might be and order processing instance waiting for an automated system to enter credit card information, and the automated system is not operational. In this case, you can modify a data instance to add credit card information manually and progress the order for a customer.
- to remove erroneous flows to tidy up your system. You might need to do this if you have been developing new flows and want to remove older versions, which are no longer required.
- where you want to search for all flow instances over a certain age and remove them. You can also do this using the instance cleaner parameters tools (see section Engine Flow and Data Instance Cleaner Parameters on page 257); however, you might not want to remove all the flow instances, just for one specific flow.
- to delete a orphaned Data definition that is not linked to a Flow definition, or perhaps where the Data definition was linked to a Flow that has been deleted.
- to test your OVBPI system. You might want to delete all the active Data definitions to reset the system.

- to change the status of services that are not behaving as they should be.
- reset the totals and averages for Flow and Data definitions. You might want to do this periodically (daily, weekly, monthly) so you can see the totals and averages in terms of a known period.

Accessing the Intervention Client

You access the Intervention Client as follows:

1. Open a new Web browser window
2. Type the following URL:

   `http://`*`hostname`*`:44080/ovbpiintclient/index.jsp`

   where:

   — *`hostname`* is the fully qualified hostname of the system where the OVBPI Server is running.

   — `44080` is the port number for the Servlet Engine, identified by the `ServletEngine HTTP` port number. Use the port number configured for your system.

   If you are running the Web Browser on the same system as the OVBPI Server, you can use `localhost` in the URL.
3. You are presented with a dialog to enter the login credentials for the Intervention Client. Enter a User Name and Password for the Client. By default the User Name is `admin` and the Password is `ovbpi`. Section Security and the Intervention Client on page 256 describes how to change the username and password credentials for the Intervention Client.

Using the Intervention Client

On the home page for the Intervention Client, you are offered the following menu options:

- Flow Definitions
- Data Definitions
- Services

Typically, you have identified an anomaly using the Business Process Dashboard, and are therefore using the Intervention Client to solve a specific problem. In this case, you have the identifier of the definition that you want to modify and you can enter this identifier directly on the search page to access the definition that you want to modify. Alternatively, you can use the Intervention Client's filtering capability to search for the instance that you want.

When you select one of the options (Flow Definitions, Data Definitions or Services), you are presented with a list of definitions within the selected category. The actions that you can complete at this point depend on the definition that you have selected:

- for Flow definitions, refer to section Flow Definitions on page 254.
- for Data definitions, refer to section Data Definitions on page 255.
- for Services, refer to section Services on page 256.

⚠ If you make changes to the values of data properties within a flow, the progression rules associated with the data properties are evaluated and the flow progresses, based on the new values.

You need to be aware of the following when using the Intervention Client:

- When selecting the option to delete a Flow and associated Data definition, make sure that the Data definition is not also a dependency for other Flow definitions. If the Intervention Client detects that by deleting the Data definition other deployed flows are affected, it presents you with a page that lists all the flows that also depend on the Data definition, where appropriate.
- When you select the option to delete a data instance, or a flow instance including all its data instances, you can potentially leave other flow instances in an indeterminate state.

  This is a similar scenario to the previous bullet describing Flow and Data definitions; however, in the case of instances, the Business Impact Engine cannot determine any relationships between the data instance that is being deleted and other flow instances that might have a dependency on it.

  If you do delete a data instance and there are flow instances that depend on it, the flow can no longer progress.

If you want to delete a significant number of flow instances or data instances, the quickest way to achieve this is to delete the Flow definition or Data definition; however, this deletes all instances of the Flow and Data definitions, including any active instances. You can then redeploy the Flow and Data definitions using the OVBPI Modeler.

Flow Definitions

When you select the Flow Definitions option from the Intervention Client menu, you are presented with a list of Flow definitions that are currently superseded and Flow definitions that are deployed in the Business Impact Engine.

The Flow definitions within the Modeler are not impacted by the actions of the Intervention Client, so you can redeploy the Flow; however active and completed instances of the flow are potentially impacted by changes made through the Intervention Client.

At this stage, you can choose on of the following options:

- Delete Flow Definition and instances
- Delete Flow and Data Definition and all related instances
- Clear Average Time and Total Instance Count (Flow and Nodes)

  This clears (or removes) the values for TotalFlows and AvrgTime from the Flows table in the Business Impact Engine database. Appendix A, Database Schemas describes the database tables in detail.

Alternatively, you can select the `Search` option to identify specific instances of the definition.

From the search page, you can enter a Flow instance identifier directly, or you can filter for instances according to the following criteria:

- flow instances with a specific identifier and value for the Weight parameter.
- flow instances within specified time periods.
- flow instances that are in a particular state, for example: `Active` or `Completed`.

When you have selected your search criteria, click the `Search` button and you are presented with a `Flow Instance List`, where all the flow instances matching your search criteria are listed. If no criteria are listed, all Flow instances are returned.

From this list, you can select instances from the list and delete them as required.

In addition, you can edit a selected Flow instances. To do this select the `Edit` option, and a flow diagram is displayed. The flow diagram shows the status of each node in the flow, for example, whether it is in the started or completed state. You can progress flow instances from this option, by manually activating or completing nodes in the flow.

There are three icons:

- the tick indicates that the node instance has completed at least once
- the cog wheel indicates that the node instance is started
- no icon indicates that the node instance is not started or completed

Data Definitions

When you select the Data Definitions option from the Intervention Client menu, you are presented with a list of Data Definitions that are currently deployed in the Business Impact Engine.

At this stage, you can choose one of the following options:

- Delete Data Definition and instances
- Clear Average Time and Total Instance Count

  This clears (or removes) the values for TotalInstances and AvrgTime from the Data Objects table in the Business Impact Engine database. Appendix A, Database Schemas describes the database tables in detail.

Alternatively, you can select the `Search` option to identify specific instances of the definition.

From the search page you can filter based on the values of the properties of the Data instances.

Following a request to search for a specific Data instance, you are presented with a `Data Instance List` page where you can delete or edit the Data Instances that match the search criteria.

When you choose to edit a Data instance, you are presented with a `Data instance` screen where you have access to the properties of the Data instance in order to confirm that it is the definition that you are interested in.

When modifying the properties of a Data instance, make sure that you enter the correct `Type` for the property, for example, if the property is an Integer, a numeric value for the property must be specified.

The Data definitions within the Modeler are not impacted by the actions of the Intervention Client, so you can redeploy the Data definition; however all active instances of the Data definition, and possibly the associated Flow definition, are impacted by changes made through the Intervention Client.

Services

When you select the Services option from the Intervention Client menu, you are presented with a list of services that are currently deployed in the Business Impact Engine. From this Window you can edit a specific service by selecting the `Edit` option. From the Service screen you can modify the status of a service.

Any changes that you make affect only the OVBPI system, the changes have no effect on the Service as it might be defined in OVO or OVIS.

Security and the Intervention Client

The Intervention Client requires you to provide a username and password in order to access the administration screens. Following a new OVBPI installation the username and password are:

- Username: `admin`
- Password: `ovbpi`

You can modify these credentials using the Tomcat Realm configuration. Appendix B, Servlet Container Authentication describes the `tomcat-users.xml` file that you can edit to change the login credentials for the Intervention Client.

Engine Flow and Data Instance Cleaner Parameters

The Engine Instance Cleaner parameters enable you to control the numbers of Completed and Active Flow and Data instances that are stored in the database. There are separate parameters for deleting Completed and Active Metric instances from the database; see section Metric Engine Instance Cleaner Parameters on page 264.

Using the Engine instance cleaner parameters, you can control:

- how often the Engine Instance Cleaner thread marks Completed and Active Flow and Data instances as candidates to be deleted.
- how often the Engine Instance Cleaner thread is run
- how often active and completed flow and data instances are deleted from the database (if at all)
- the time and the age of the instances that you delete

Following a new OVBPI installation, the Engine Instance Cleaner thread does not run and Completed and Active instances therefore accumulate in the database.

This section describes the behavior of these parameters and the SQL procedures, which you can modify to enable you to archive the data, before it is deleted, to a location of your choice. By archiving the data you have it available for your reporting tools, but it is no longer stored in the OVBPI database tables and therefore does not impact the performance of your system.

If you configure these parameters to delete Completed and Active instance data from the database, and you do not archive the instance data, you cannot customize the OVBPI Business Process Dashboard to monitor Completed and Active instances.

In summary, the instance cleaner parameters are intended for regular removal of Completed and Active instances as they occur during normal operations. They are not intended to be used for the removal of individual completed instances, or removal of instances in states other than the `COMPLETED` or `ACTIVE` state. Use the Intervention Client to remove individual instances; see section Intervention Client on page 251.

Configuring the Engine Instance Cleaner Parameters

You modify the instance cleaner parameters using the OVBPI Administration Console as described in section Engine Flow and Data Instance Cleaner Settings on page 83.

These parameters enable you to mark instances as being candidates to be deleted. Instances are initially marked to be deleted and not immediately deleted to reduce the impact of the database activity on other Business Impact Engine activities. If you are using a remote database, this might not be an issue for your implementation. Once the instances are candidates to be deleted, they are deleted periodically by the instance cleaner thread as specified by the instance cleaner parameter values. You can specify the frequency that you want them deleted and the numbers of instances that you want deleted in one database Delete statement, using the parameters.

In addition to modifying the instance cleaner parameters, you can modify the stored procedures invoked by the instance cleaner thread and add SQL commands to archive the Business Impact Engine data to a location of your choice; see section Archiving Completed and Active Flow Instances on page 261.

The following are some examples of how you can set the instance cleaner parameters to achieve specific outcomes, or requirements.

Example for Deleting Completed Instances

If you want to delete complete flow instances from the database that are more than one day old and once a day, set the instance cleaner parameters as follows:

- From the `Engine Flow and Data Instance Cleaner settings:`
  - — Set the `Mark instances for deletion:` option to `Once a Day`.

    Enter `01:00` as the time for the instances to be marked. Pick a time when the database is not busy processing other OVBPI data.
  - — Set the Delete interval: option to 1 minute.

    This is the interval at which the Flow and Data instance cleaner thread runs and attempts to delete any Flow and Data instances that are marked as being candidates to be deleted.
  - — Set the Delete batch size to 500

    This is the number of instances that the instance cleaner thread attempts to delete each time it runs.
- From the `Completed Flow and Data Instances settings:`
  - — Set the `Delete completed instances from the database` option to `As scheduled above`

    The Completed instances are then deleted on the basis of the times set for the Engine Instance Cleaner thread.
  - — Set the `Age of the completed instances to be removed (minutes)` to two days, which is expressed in minutes (2880 minutes).

Example for Deleting Active and Completed Instances

If you want to delete Completed flow instances that have an age of more than 30 days, and Active flow instances that have an age of more than 50 days, once a day, set the instance cleaner parameters as follows:

- From the `Engine Flow and Data Instance Cleaner settings`:
  - Set the `Mark instances for deletion:` option to `Once a Day`.

    Enter `01:00` as the time for the instances to be marked. Pick a time when the database is not busy processing other OVBPI data.
  - Set the Delete interval: option to 1 minute.

    This is the interval at which the Flow and Data instance cleaner thread runs and attempts to delete any Flow and Data instances that are marked as being candidates to be deleted.
  - Set the Delete batch size to 1000

    This is the number of instances that the instance cleaner thread attempts to delete each time it runs.
- From the `Completed Instances settings`:
  - Set the `Delete completed instances from the database` option to `As scheduled above`
  - Set the `Age of the completed instances to be removed (minutes) to` 30 days, which is expressed in minutes (43200 minutes).
- From the `Active Instances settings`:
  - Check the `Delete active instances from the database?` check box so the active instances are deleted on the basis of the Engine Instance Cleaner thread settings.
  - Set the `Age of the active instances to be removed to` 50 days.

Example for Deleting Completed Instances as Soon as They Are Complete

If you want to delete Completed instances as soon as they are complete, set the instance cleaner parameters as follows:

- Set the `Engine Flow and Data Instance Cleaner settings` as required for the Active instances, as these settings are ignored in the case of the Completed Instances setting `Immediately on completion`.
- From the `Completed Instances settings`:
  - — Set the `Delete completed instances from the database` option to `Immediately on completion`
  - — `Age of the completed instances to be removed (minutes)` is also not applicable and is not an editable field.

  Be aware that using this setting has the effect of removing data that might have a value for monitoring purposes. If your flow instances do not have long lives, then there is going to be very little data in the database to be reported on. Make sure that you do not need the data that you are deleting from the database.
- Set the `Active Instances settings` as required.

Archiving Completed and Active Flow Instances

You have the option to archive Completed and Active instances before they are deleted from the database by the instance cleaner thread. This enables you to remove data from the OVBPI database to improve performance, but still keep the data for reporting purposes.

➤ You cannot archive Completed instances if you choose the option to `Delete completed instances from the database Immediately on completion`. This is because for this option, the instances are deleted as soon as the event that completes the instance is received.

To archive the instance data, you need to make changes to the stored procedures or functions provided with OVBPI to delete the flow and data instances. There are stored procedures for Microsoft SQL Server and functions for the Oracle Server, both contain the SQL commands required to delete data from the OVBPI database and also provide a return code to the calling program. The following are the stored procedures/functions:

- `bia_DeleteDataInstances`

  This procedure deletes the data instances in the `COMPLETED` state and that meet the criteria specified in the configuration. This stored procedure also provides the following SQL Cursor that you can use:

  `completedInstances_cursor`

  This stored procedure is available for Microsoft SQL Server only.

  You can add your custom SQL statements to the stored procedure in order to archive the business flow data before it is deleted.

- `bia_DeleteFlowInstances`

  This procedure deletes all flow instances in the `COMPLETED` state that meet the criteria specified by the configuration. This stored procedure also deletes the associated node instances, node started and node completed times, and any metrics that are defined for the flow instance.

- `DeleteActiveDataInstances`

  This procedure deletes the data instances in the `ACTIVE` state and that meet the criteria specified in the configuration. This stored procedure also provides the following SQL Cursor that you can use:

  `activeInstances_cursor`

  This stored procedure is available for Microsoft SQL Server only.

  As for the `bia_DeletedDataInstances` procedure, you can add your custom SQL statements to the stored procedure in order to archive the business flow data before it is deleted.

- `DeleteActiveFlowInstances`

  This procedure deletes all flow instances in the `ACTIVE` state that meet the criteria specified by the configuration. This stored procedure also deletes the associated node instances, node started and node completed times, and any metrics that are defined for the flow instance.

These stored procedures or functions are contained in the following SQL file:

`OVBPI-install-dir``\misc\bia\EngineStoredProcedures_script.sql`

There are also examples of this file located at:

- *OVBPI-install-dir*`\examples\bia\EngineScripts`
- the `examples` directory on the distribution media

The example files are named as follows, according to the database:

- `EngineStoredProcedures_script.mssql`
- `EngineStoredProcedures_script.oracle`

Be aware that changes to the SQL file `EngineStoredProcedures_script.sql` are lost if you reinstall or upgrade OVBPI. Make sure that you have made backup copies of your modifications, which you can then reapply following a reinstallation.

You need to make your changes to the stored procedures, which have been created in the database for OVBPI, to add the SQL to archive the Completed or Active instances.

You are also advised to make changes to the following SQL script and make a backup copy of the script to ensure that you have a copy of the changes:

OVBPI-install-dir`\misc\bia\EngineStoredProcedures_script.sql`

The instance cleaner parameters are executed at the frequencies specified in the `Engine Instance Cleaner Settings`. The value of the `Age of the completed instances to be removed` and `Age of the active instances to be removed` parameters is passed to the stored procedure, or function, when it is called.

It is possible for two data instances to be created, each representing the same business flow: one in the Business Impact Engine database and one in the archive database. This occurs when an out-of-sequence event is received by the Engine after the Completed or Active instances are archived and deleted. An out-of-sequence event is where an event is received by the Business Impact Engine for an activity in a flow instance that is earlier than the latest event received for the same flow instance.

In summary, you need to ensure that your archive code can take account of the fact that there might be more than one instance that represents the same business flow when this scenario exists.

Metric Engine Instance Cleaner Parameters

The Metric Engine Instance Cleaner parameters enable you to control the numbers of metric instances that are stored in the database. These include:

- Active metric instances
- Completed metric instances
- Metric statistic instances
- Metric alarm instances

There are separate parameters for deleting Completed and Active business impact Engine instances from the database; see section Engine Flow and Data Instance Cleaner Parameters on page 257.

Using the Metric Engine instance cleaner parameters, you can control:

- how often the Metric Engine Instance Cleaner thread is run.
- how often active and completed metric instances are deleted from the database (if at all).
- how often metric statistics settings are deleted from the database (of at all).
- how often metric alarm settings are deleted from the database (of at all)
- for all metric instances, the time and the age of the instances that you delete.

Following a new OVBPI installation, the Metric Engine Instance Cleaner thread does not run and the different metric instances therefore accumulate in the database.

Configuring the Metric Engine Instance Cleaner Parameters

You modify the Metric Engine instance cleaner parameters using the OVBPI Administration Console as described in section Metric Engine Instance Cleaner Settings on page 98.

These parameters enable you to delete the Metric Engine instances periodically, using the instance cleaner thread, as specified by the instance cleaner parameter values.

You can control how often the metric instance cleaner thread is executed and the age of the specific metric instances that you want to delete. Each type of metric instance can be separately controlled.

The following are some examples of how you can set the Metric Engine instance cleaner parameters to achieve specific outcomes, or requirements.

Example for Deleting Completed Metric Instances

If you want to delete completed metric instances from the database, which are more than one day old and once a day, set the metric engine instance cleaner parameters as follows:

- From the `Metric Engine Instance Cleaner settings`:
  - Set the `Instance cleaner execution interval (minutes)` to 5.
- From the `Completed Metric Instances settings`:
  - Select the `Delete completed metric instances from the database?` option in order to activate the `Age of completed metric instances to be removed (days)` option.
  - Set the `Age of completed metric instances to be removed (days)` option to 2.
- Set the other option as appropriate for your implementation.

11

Backup and Recovery

This chapter describes the backup and recovery procedures for OVBPI.

It is important to back up your OVBPI system to ensure that the OVBPI files and data can be recovered in the event of a media corruption. It is also important to back up your system to ensure that any flows being processed and their associated data, services, and events can be recovered to a known state.

There are general high reliability configurations, which are recommended for all systems, for example, using mirror or Redundant Array of Inexpensive Disk (RAID) storage. Restoring or recovering from a backup should be a last resort. Your goal should be to make sure that the backups are never required.

This section does not cover how to set up your system for high reliability. The section focuses on the recommended procedures for backing up and recovering OVBPI data. The section also describes the behavior of the OVBPI system when specific components are shutdown, and provides recommendations for minimizing the effect of the shutdown.

If you have a system shutdown or failure, you want to keep your OVBPI system as synchronized as possible during the shutdown period. This section describes techniques for design that can help achieve this.

The following list describes some key considerations that you need to be aware of when designing your OVBPI system for effective backup and subsequent recovery:

- For planned component shutdowns:
  - Shut your OVBPI system down during quiet periods where possible.
  - Use high reliability tools where appropriate.
- Design adaptors to minimize the number of events lost during a system failure.

  For most adaptors, it is possible to design them such that they process events from the last time the adaptor was executed. The *OpenView Business Process Insight Integration Training Guide - Business Adaptors* describes how you do this.
- Minimize the number of Check Sequence arcs that you use in your flows.

  When the OVBPI system is recovered following a shutdown, it is likely to receive a burst of out-of-sequence events. You are recommended to use Check Sequence arcs only where essential. Overuse can cause the OVBPI system to generate an excess of alerts when it restarts. The *OpenView Business Process Insight Concepts and Modeling Flows* describes Check Sequence arcs in more detail.

Files and Directories to Backup

The following list describes the OVBPI data that is persisted within your OVBPI system and that therefore needs to be part of your backup policy:

- All OVBPI schema objects in the database.

  You specified the database details when you installed OVBPI, you need these details in order to complete your backup.

  The database for OVBPI should be backed up using the utilities and procedures provided as part of your database management system.

- Event definitions

  OVBPI stores details about Event definitions that have been deployed to the Business Impact Engine in a series of files; these files are located as follows:

  `OVBPI-install-dir\data\repository\events`

- Configuration information

  OVBPI includes configuration information, which controls how OVBPI is setup and is relatively static. This configuration information is located in the following directory:

  `OVBPI-install-dir\data\conf\bia`

- Template configuration files

  The following directory contains the template definitions for most of the configuration files for the OVBPI Server and presented (managed) within the Administration Console. When you configure a component using the Administration Console, all the configuration files in `data\conf\bia` are regenerated, based on the content of the template files located at:

  `OVBPI-install-dir\newconfig`

- Log files

  These do not need to be backed up to restore a consistent system; however, you are advised to back up the log files, which are located at:

  `OVBPI-install-dir\data\log`

- Data files

  These are the files (Class files, Java files and so on) that contain the details of the deployment status of your definitions. OVBPI includes configuration information relating to the flows and data that are deployed within the database. This information changes whenever the components of a flow are deployed to the Business Impact Engine. This information might be static in a production system, but variable in a development system where business flows are being designed, developed and tested and are therefore in a state of change.

  These data files are located at:

  `OVBPI-install-dir\data\datafiles`

- Business Event Handler

  Events received by the Business Impact Engine from the Business Event Handler are not acknowledged by the Engine until they have been successfully received. Out-of-sequence events and events that contain errors (for example, missing data) are stored in the Business Event Handler hospital.

  The Business Event Handler includes database tables for the hospital and event store. By default these tables are created as part of the same database schema as the rest of the OVBPI data, and are therefore backed up with the OVBPI data.

  For more information about the OVBPI database schemas and tables, refer to Appendix A, Database Schemas.

  Deployed Event definitions are located, as text files, within the OVBPI installation directory as follows:

  `OVBPI-install-dir\data\repository\events`

- Application adaptor files and data

  Application adaptors are those developed using the Business Event Handler component to access business data from applications within your organization. The *OpenView Business Process Insight Integration Training Guide - Business Events* describes how to build application adaptors.

  It is usual for application adaptors to be located close to (or co-located with) their data source, which means that they are likely to be running on a different system to the system where OVBPI is installed.

  Any data used by that application adaptor that needs to be synchronized with the application data (for example buffer tables populated using database triggers or adaptor history files), should be backed up as part of the backup procedures for the application. Where possible the buffer table should exist within the same database as the application data that is being monitored. This makes it easier to backup the data together and keep it synchronized.

- OVBPI OVO Adaptor

  Configuration files for the OVBPI OVO adaptor are located on the system where the adaptor and OpenView Operations are installed.

  For HP-UX, these files are located at:

  ```
  OVBPI_install_dir/data/conf/bia
  OVBPI_install_dir/misc/bia
  ```

  For Windows, these files are located at:

  ```
  OVBPI_install_dir\data\conf\bia
  OVBPI_install_dir\misc\bia
  ```

  Log files for the OVO adaptor are also located on the system where OpenView Operations is installed as follows:

  `OVBPI-install-dir/data/log` (HP-UX)

  `OVBPI-install-dir\data\log` (Windows)

With the exception of the application adaptor files and data, you must back up all of the data described above at the same time in order to maintain its consistency. The OVO adaptor data must be backed up when the application data is backed up.

How often you back up the OVBPI data depends on how frequently the data is changing. The data can be backed up while the OVBPI system is operational; however, you need to ensure that you do not deploy or undeploy and definitions during the backup.

As the OVBPI database cannot be synchronized with the Event definition files and data files, it is not possible to recover the OVBPI system to the point of failure from a backup. You must use the appropriate database management system tools to restore your system to the most recent backup and not to the point of failure.

As you cannot recover to the point of failure, if you have installed a database solely for OVBPI, you can consider not backing up all the transaction log files. This will reduce the amount of disk space required by the database for recovery purposes. In the case of Microsoft SQL Server, you could use the Simple Recovery model, rather than the Full Recovery model. The Simple Recovery model enables you to recover to the most recent backup. The Full Recovery model enables you to recover to the point of failure.

In the case of Oracle, you need to use ARCHIVELOG to enable online backups. If you do not want to use ARCHIVELOG, you can stop OVBPI and complete an offline backup.

How the OVBPI System Behaves When Components are Shutdown

This section describes the behavior of the OVBPI components when they are shutdown. It provides additional information that helps you understand the behavior of the system for backup and recovery purposes.

Engine Shutdown

If the Business Impact Engine component is shutdown, it is no longer able to receive and process events. In addition, the status of OVIS and OVO Services that you have linked to from OVBPI are no longer updated.

As a result of the Engine shut down, the Business Event Handler component stops accepting business events until the Business Impact Engine is restarted. When the Engine is restarted, any business events waiting to be processed are automatically submitted to the Business Impact Engine.

Database Shutdown

If the database is shut down for any reason, the Business Impact Engine can no longer communicate with it and also shuts down, as described in section Engine Shutdown on page 273. In addition, all other OVBPI components that use the database are shut down.

Notification Server Shutdown

When the OVBPI system determines that a notification is required, the Business Impact Engine passes a notification alert to the Notification Server. All notification alerts are held by the Business Impact Engine, in the database, until the Notification Server has confirmed that it has accepted responsibility for them.

If the Business Impact Engine shuts down before the alert is sent to the Notification Server, the alert is retried when the Business Impact Engine restarts and the service status is shown as usual through the Business Process Dashboard. Note that the service status shown is the latest status. No notifications are generated for any interim changes that occur when the

Business Impact Engine is shut down. For example, if the Engine shuts down when a service is in the state `Critical`, and the service subsequently changed from `Critical` to `Normal` and then back to `Critical`, when the Engine restarts, the service status is shown as `Critical.` In this case, the information about the state change to `Normal` and then back to `Critical` is not recorded.

Email Server Shutdown

If the Notification Server cannot make a connection to the email server to send the alert to the designated user, the Notification Server stores the alert in the database and resends it at a later time when a connection is established. The Notification Server continues the attempt to resend the alert until is reaches the value for the `Maximum retries to SMTP server parameter`. When this maximum is reached, the Notification Server stops retrying and logs an error in its log file.

OVBPI OVO Adaptor Shutdown

If the OVBPI OVO Adaptor is shutdown for any reason the OVBPI system can no longer receive operational events from OpenView Operations, nor can it deploy flows that depend on services that are monitored using the adaptor.

When the adaptor is restarted, the Business Impact Engine queries the current state of all the services it is monitoring, which means that no operational events are lost. The latest state of the service is reported, but any changes in state while the OVBPI system was shut down are not reported.

Business Event Handler - OVBPI Adaptor Shutdown

If the Business Event Handler shuts down for any reason, the effect of the shutdown depends on the design of the individual adaptors that are connected to it.

For database adaptors, where you are using buffer tables, make sure that the buffer tables are backed up and synchronized with the application data that is being monitored.

For file adaptors, make sure that the current position in the file is backed up with the source application data.

This means that when the Business Event Handler OVBPI adaptor is restarted, the application adaptors can be also restarted and any events marked as `DISCHARGE_WAIT` can then be sent back into the OVBPI system.

Servlet Engine Shut Down

If the Servlet Engine is shut down for any reason, all the currently open OVBPI clients using Browser Windows also stop. You might need to restart the Web Browser Windows and the OVBPI clients when the Servlet Engine is restarted.

Completing an Online Backup for OVBPI

Before starting an online backup, you need to ensure that your database is configured such that it is capable of completing an backup online. For example, in the case of an Oracle Server, you need to make sure that the Oracle instance you have specified for OVBPI is running in `ARCHIVELOG` mode. If it is not, you cannot complete an online backup.

The following steps describe the process for completing an online backup:

1. Ensure no one is using the OVBPI Modeler.
2. Close the OVBPI Modeler.
3. Use the OVBPI Administration Console to shut down the Model Repository component. This prevents changes being added to the repository whilst the backup is taking place.

   Do not make any further administration changes using the Administration Console until after the online backup is complete.
4. Make sure that no changes are made to the OpenView Operations adaptor configuration during the online backup.
5. Backup the following OVBPI directories and their subdirectories on the system where the OVBPI Server is installed:

   ```
   OVBPI-install-dir\data\repository
   OVBPI-install-dir\data\conf\bia
   OVBPI-install-dir\data\datafiles
   OVBPI-install-dir\data\log
   OVBPI-install-dir\newconfig
   ```

   If you prefer, you can backup the entire *`OVBPI-install-dir`*`\data` directory.

➤ Any log files currently locked by OVBPI processes cannot be backed up, so there are some error messages in the backup process referring to these files. Older log files are successfully backed up.

6. Back up the OVBPI data in the database using the database management system backup utilities.

7. Backup the following OVBPI directories and their subdirectories on the system where the OVO adaptor is installed:

   ```
   OVBPI-install-dir\data\conf\bia
   OVBPI-install-dir\misc\bia
   OVBPI-install-dir\data\log
   OVBPI-install-dir\newconfig
   ```

8. Use the OVBPI Administration Console to restart the Model Repository component on the system where the OVBPI Server is installed.

9. Restart the OVBPI Modeler.

Your backup is now complete.

It is possible that the Engine Model Cleaner process is running at the same time as the online backup is completed. This is not a problem, but can result in inconsistencies between the data stored in the Engine's database and its configuration files. If you see warnings in the Business Impact Engine log file indicating that the Engine cannot find a definition in the database, it is as a result of this scenario.

Restoring your OVBPI Data from a Backup

This section describes how to restore your OVBPI system following:

- a complete system failure
- an OVBPI file system failure
- a database failure

In all cases, you are restoring both the database and the file system for the OVBPI components.

The following are points to note when restoring the database and the file system:

- Flow instances created after the last database backup are lost.
- Flow instances completed after the last database backup remain active and are unlikely to complete.
- Flow, Data, Service or Event definitions created since the last database backup are lost and need to be recreated.
- Modifications to Flow, Data, Service or Event definitions since the last database backup are lost and need to be reapplied.
- Flow, Data, Service or Event definitions deployed since the last database backup are lost and need to be redeployed.
- Metric definitions that are new or have been modified since the last database backup need to be recreated or reapplied.
- Email users and subscriptions configured through the Notification Server Administration Console after the last backup are lost and need to be re-added.
- Scripts and files created since the last backup need to be recreated; for example, any Velocity templates or Notification Server scripts that you have recently created might need to be redone.
- Email users and subscriptions deleted since the last database backup reappear and will need to be deleted again.
- Flow, Data, Service and Event definitions that have been undeployed reappear and need to be undeployed again.

Where appropriate, you are advised to restore the database files, OVBPI Event definition files and OVBPI configuration files to maintain consistency. If you do this, you also need to re-enter any Event definitions created since the backup was completed.

Steps to Restore your OVBPI System

This section describes the steps for restoring your OVBPI system.

If you have had a severe system failure, you are recommended to reinstall OVBPI including all its dependencies, for example, the database management system (Microsoft SQL Server or Oracle Server) and the J2SE, before restoring any files.

To restore the OVBPI data, complete the following steps:

1. Ensure no one is using the OVBPI Modeler.
2. Close the OVBPI Modeler to prevent new and modified definitions being created.
3. Use the OVBPI Administration Console to shut down all the OVBPI components.

   Do not make any further administration changes using the Administration Console until after the recovery is complete.
4. Restore the files that you previously backed up under:

   ```
   OVBPI-install-dir\data
   OVBPI-install-dir\newconfig
   ```

   If your OVBPI system has been running since the last backup you are likely to receive warnings for log files that already exist; this is expected. However, other error and warning messages should be fully investigated before continuing.

   If you are using an Oracle database continue at step 6. If you are using a Microsoft SQL Server database, continue at step 5.

5. Use the Microsoft SQL Server database utilities to restore the OVBPI database to the last backup. Be aware that when restoring files following the reinstallation of the database, the system identifiers for the database users are not the same. As a result, the newly created OVBPI user is not able to access the data in the restored database.

   For Microsoft SQL Server you use the Query Analyzer to execute the an SQL command similar to the following (if you are not using the default schema name or database user defined within OVBPI, substitute the correct schema name or user for your database):

   ```
   use OvbpiSchema;
   exec sp_change_users_login ‘Update_One’, ‘ovbpiuser’,
   ‘ovbpiuser’;
   ```

   You also need to ensure that the default database for your database user is set correctly; for example: `OvbpiSchema`.

   ➤ As the OVBPI database cannot be synchronized with the OVBPI file data relating to Event definitions and configuration information, it is not possible to recover to the point of failure from a backup. You must use a database management system command to restore your system to the most recent backup and not to the point of failure.

   Continue at step 7.

6. Use the Oracle Server database tools to restore the OVBPI database to the last backup.

   ➤ As the OVBPI database cannot be synchronized with the OVBPI file data relating to Event definitions and configuration information, it is not possible to recover to the point of failure from a backup. You must use a database management system command to restore your system to the most recent backup and not to the point of failure.

7. Restart the Business Impact Engine, Metric Engine and Model Repository from the OVBPI Administration Console.
8. Restart the OVBPI Modeler.
9. Open a Metric definer Window.

10. Re-apply any business flow data that has been entered through the OVBPI Modeler, since the last backup.
11. Re-apply any metric definition data that has been entered using the Metric definer, since the last backup.

    You have now restored your OVBPI data.
12. Undeploy any Flow, Data, Service or Event definitions that are deployed and that were undeployed before you restored your OVBPI system.
13. Use the OVBPI Administration Console to restart the remaining OVBPI components.
14. Delete any email users and subscriptions that have reappeared since you restored your OVBPI system.

Now go to section Completing the OVBPI Recovery Procedure on page 282 to complete the recovery procedure.

Completing the OVBPI Recovery Procedure

This section describes points to note after you have completed an OVBPI recovery from backup.

After you have completed the steps described in section Restoring your OVBPI Data from a Backup on page 278, you have recovered to the latest backup. You have therefore potentially missed events that are relevant to your business flows. This means that OVBPI can raise erroneous events, particularly in cases where you have Check Sequence arcs defined.

You do also have the option to take a differential backup, which is faster. However, you are advised to adopt the backup and recover policies adopted for other applications within your organization, using the information provided in this section, which is specific to 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 and Business Events Handler.

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 286 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 317.

- 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 334.
- 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 338 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 49 Oracle and SQL Data Types

Descriptive Type	Oracle Type	SQL Server Type	Description
string	varchar2	nvarchar	Variable length character field.
date[1]	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`).

1. 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 317, 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.

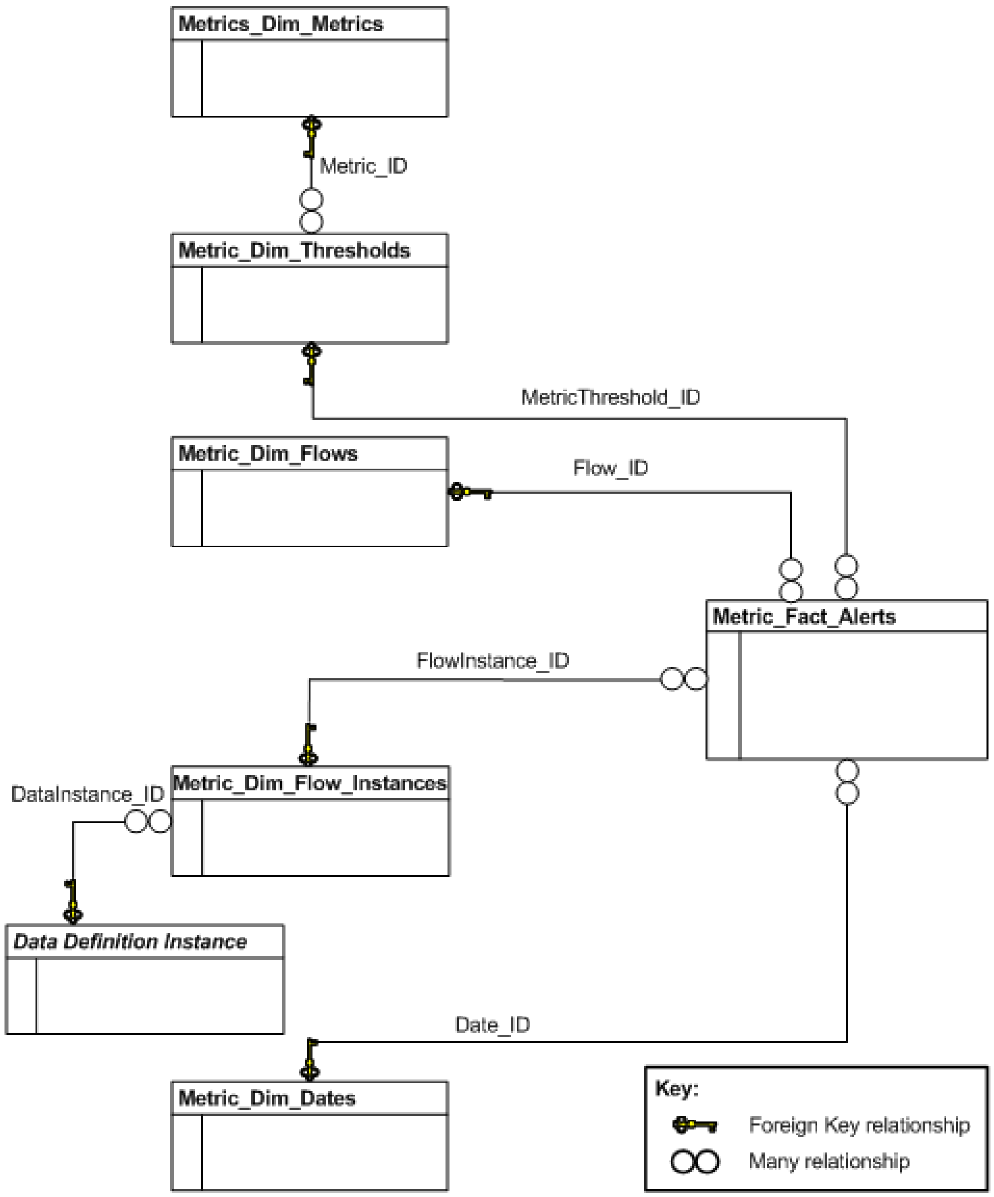
The following sections describe the components that make up the Business Metric schema. The data types listed in the tables are the SQL data types, Table Oracle and SQL Data Types on page 285 describes the SQL Server and Oracle data types mappings and their descriptions.

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 336 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 299.

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

Table 50 Metric_Fact_Alerts

Column Name	Data Type	Length	Allow Nulls	Description
MetricAlert_ID	string	36	No	Primary Key^1.
MetricThreshold_ID	string	36	No	Foreign Key^2 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 50 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: • 1 = `Normal` • 2 = `Warning` • 3 = `Minor` • 4 = `Major` • 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.

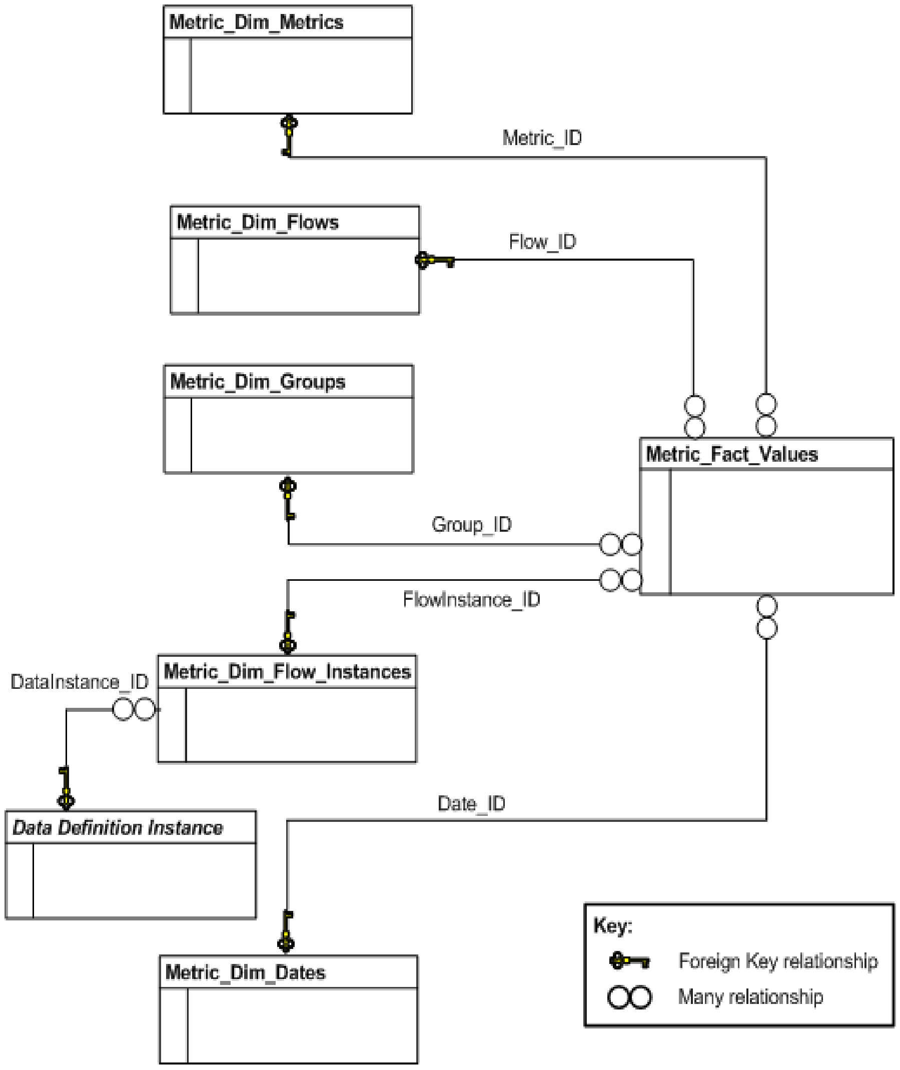
1. This is the column that uniquely identifies rows.
2. This is the column that exists as the primary key in another table.

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 336 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 299.

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 51 Metric_Fact_Value

Column Name	Data Type	Length	Allow Nulls	Description
MetricValue_ID	string	36	No	Primary Key[1].
Metric_ID	string	36	No	Foreign Key[2] 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 51 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: • `Active` - the start condition for the metric has been met, but the end condition has not been met. • `Completed` - the metric has completed and has a value for reporting. • `NoStart` - the metric has completed, but does not have a value. • `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. 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 51 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.

1. This is the column that uniquely identifies rows.

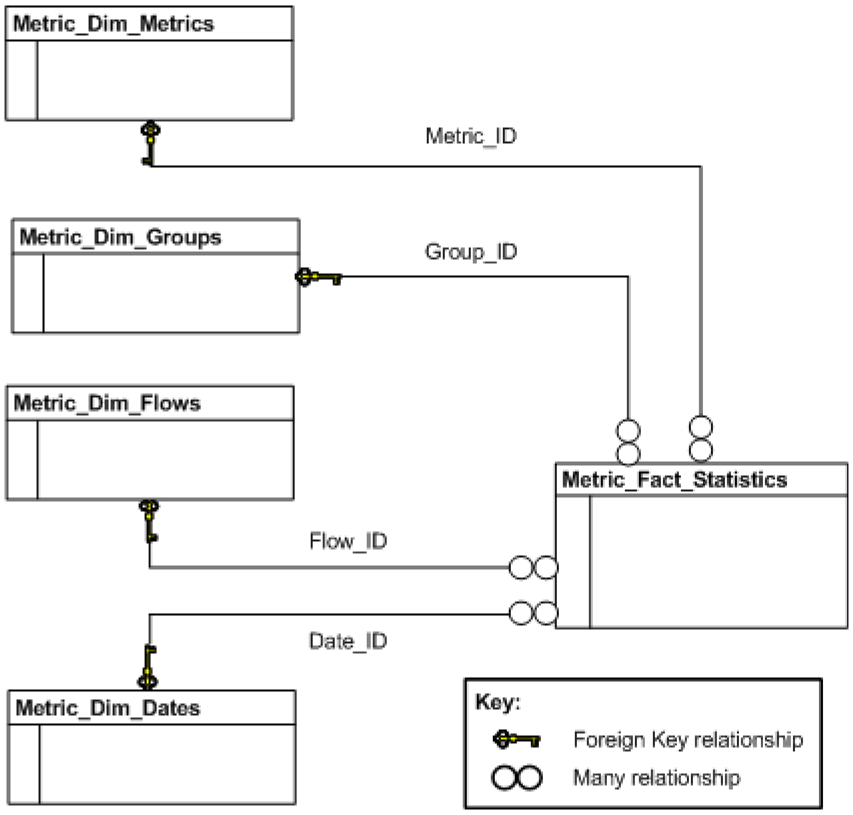
2. This is the column that exists as the primary key in another table.

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 299.

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 52 Metric_Fact_Statistics

Column Name	Data Type	Length	Allow Nulls	Description
MetricStatistic_ID	string	36	No	Primary Key[1].
Metric_ID	string	36	No	Foreign Key[2] 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 52 Metric_Fact_Statistics

Column Name	Data Type	Length	Allow Nulls	Description
StatisticsType	string	12	Yes	The type of statistic: • `Active` - all currently active metric instances that relate to this statistic. • `Completed` - all currently completed metric instances that relate to this statistic. • `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. 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. The number of metric instances per hour. This is calculated as follows: `(CountTotal * 36,000)/ MeasurementPeriod`
SumValue	float		Yes	Total for the metric instance values.

Table 52 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. This is number of instance per hour that relate to this metric. The value is calculated as follows: `(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 52 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.

1. This is the column that uniquely identifies rows.
2. 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 53 Metric_Dim_Metrics

Column Name	Data Type	Length	Allow Nulls	Description
Metric_ID	string	36	No	Primary Key[1].
MetricName	string	120	No	Name given to a business process metric through the Business Process Metric definer.
Flow_ID	string	36	No	Foreign Key[2] to the Flows table; see Table 62 on page 318.
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 53 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.

1. This is the column that uniquely identifies rows.
2. 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 54 Metric_Dim_Dates

Column Name	Data Type	Length	Allow Nulls	Description
Date_ID	string	36	No	Primary Key[1]
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: • 1 = January to March • 2 = April to June • 3 = July to September • 4 = October to December

Table 54 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: • 1 = Sunday • 2 = Monday • 3 = Tuesday • 4 = Wednesday • 5 = Thursday • 6 = Friday • 7 = Saturday
Hour	integer		No	The `hour` for the time, represented as a numeral between 0 and 23.

Table 54 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.

1. 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 55 Metric_Dim_Flows

Column Name	Data Type	Length	Allow Nulls	Description
Flow_ID	string	36	No	Primary Key[1].
FlowName	string	120	No	The name of the business flow as entered in the OVBPI Modeler.

Table 55 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: • `Active` means the flow has been deployed. • `Impeded` means that at least one instance of the flow is in the `Impeded` state; see `Status` in Table 63 on page 320. • `Blocked` means that at least one instance of the flow is in the `Blocked` state; see `Status` in Table 63 on page 320. • `Deleted` means the flow has been undeployed.
DataDefinition_ID	string	36	Yes	Primary entity identifier for the Data definition associated with the Flow.

1. 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 56 Metric_Dim_Flow_Instances

Column Name	Data Type	Length	Allow Nulls	Description
FlowInstance_ID	string	36	No	Primary Key[1].
Flow_ID	string	36	No	Foreign Key[2] to table Flows (Flow_ID) column.
Identifier	string	120	Yes	The identity of the related Data definition for this Flow. 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 56 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: • `Active` means the flow instance has met the start condition for at least one node in the flow. • `Completed` means the flow instance has met the complete conditions for one of the end nodes in the flow. 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.

1. This is the column that uniquely identifies rows.
2. 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 57 Metric_Dim_Groups

Column Name	Data Type	Length	Allow Nulls	Description
Group_ID	string	36	No	Primary Key[1].
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.

1. 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 58 Metric_Dim_Thresholds

Column Name	Data Type	Length	Allow Nulls	Description
MetricThreshold_ ID	string	36	No	Primary Key[1].
Metric_ID	string	36	No	Foreign Key[2] 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 58 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: • `Instance` - set for a particular instance • `Active` - set for active instances • `Completed` - set for instances that have completed during the metric period • `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: • `CountTotal` - number of instances • `Throughput` - throughput • `AverageValue` - average • `MinimumValue` - minimum value • `MaximumValue` - maximum value • `Null`- when `ThresholdType` is `Instance` from `Value` column in `Metric_Fact_Value` table. • `WeightTotal` - weight total • `WeightThroughput` - weight throughput • `WeightAverageValue` - weight average

Table 58 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: • `OverValue` - value is equal to or more than value specified in the alert level. • `UnderValue` - value is less than or equal to value specified in the alert level. • `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) • `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) • `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. • `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 58 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: • 1 = `Normal` • 2 = `Warning` • 3 = `Minor` • 4 = `Major` • 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 58 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.

1. This is the column that uniquely identifies rows.
2. 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 59 on page 312. 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 59 Metric_CustomTypes

Column Name	Data Type	Length	Allow Nulls	Description
CustomMetric Name	string	120	No	Primary Key[1]. 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).

1. 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 60 on page 313)
- Metrics_Value view (Table 61 on page 315)

Table 60 on page 313 is a view onto the Metrics_Dim_Metrics table described in section Business Metrics Dimensions on page 299.

Table 60 Metrics

Column Name	Data Type	Length	Allow Nulls	Description
MetricName	string	40	No	Primary Key^1 and the name given to the metric through the OVBPI Modeler.
MetricType	string	6	Yes	The type of metric, for example: • NET (Single Node scope) • TBN (Multiple Node scope) • FET (Whole Flow scope) • CUSTOM (custom-written metric)
Flow_ID	string	36	No	Primary Key^1 and Foreign Key^2 to the Flows table; Table 63 on page 320.

Table 60 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. In the case of FET, `StartNode` is null. 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. In the case of FET, `StartNode` is null. 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.

1. This is the column that uniquely identifies rows.
2. 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 61 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: • NET (Single Node scope) • TBN (Multiple Node scope) • FET (Whole Flow scope) • CUSTOM (custom-written metric)
Flow_Instance	string	36	Yes	Foreign Key[1] to Flow_Instance table; see Table 63 on page 320.
StartTime	date		Yes	Time when flow instance met the Start condition for `StartNode` (in date format); see Table 60 on page 313 for a description of `StartNode`.
StartTimeLong Millis	long		Yes	Time (in milliseconds) when flow instance met the Start condition for `StartNode`; see Table 60 on page 313.
EndTime	date		Yes	Time when flow instance met the complete condition for `EndNode` (in date format); see Table 60 on page 313.
EndTimeLong Millis	long		Yes	Time (in milliseconds) when flow instance met the complete condition for `EndNode`; see Table 60 on page 313.
Value	float		Yes	For internal use.

Table 61 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.

1. 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 286. 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 333.

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 285 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 333, 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 62 Flows

Column Name	Data Type	Length	Allow Nulls	Description
Flow_ID	string	36	No	Primary Key[1].
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 62 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: • `Active` means the flow has been deployed. • `Impeded` means that at least one instance of the flow is in the `Impeded` state; see `Status` in Table 63 on page 320. • `Blocked` means that at least one instance of the flow is in the `Blocked` state; see `Status` in Table 63 on page 320. • `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 62 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.

1. 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 63 Flow_Instance

Column Name	Data Type	Length	Allow Nulls	Description
FlowInstance_ID	string	36	No	Primary Key[1].
Flow_ID	string	36	No	Foreign Key[2] to table Flows (`Flow_ID`) column.
Identifier	string	120	Yes	The identity of the related Data definition for this Flow. 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 63 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. This column contains the value of the Data Definition property, nominated to be the Weight property, in the OVBPI Modeler. 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. 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 63 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. The `Weight_type` column provides the context for the values in the Weight column. 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 63 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: • `Active` means the flow instance has met the start condition for at least one node in the flow. • `Impeded` means that the flow instance cannot progress to completion without encountering a blocked node. • `Blocked` means that the flow instance is active at a node where the node's services have failed. • `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

1. This is the column that uniquely identifies rows.
2. 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 64 Nodes

Column Name	Data Type	Length	Allow Nulls	Description
Node_ID	string	36	No	Primary Key[1].
Flow_ID	string	36	No	Foreign Key[2] 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 64 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 64 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.

1. This is the column that uniquely identifies rows.
2. 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 65 Node_Instance

Column Name	Data Type	Length	Allow Nulls	Description
NodeInstance_ID	string	36	No	Primary Key[1].
FlowInstance_ID	string	36	No	Foreign Key[2] to table Flow_Instance (`FlowInstance_ID`).
Node_ID	string	36	No	Foreign Key[2] to table Nodes (`Node_ID`).

Table 65 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[3]	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.

1. This is the column that uniquely identifies rows.
2. This is the column that exists as the primary key in another table.
3. `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 66 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 67 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 68 Arcs

Column Name	Data Type	Length	Allow Nulls	Description
Flow_ID	string	36	No	Foreign Key[1] 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.

1. 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 69 Resources

Column Name	Data Type	Length	Allow Nulls	Description
Resource_ID	string	260	No	Primary Key[1].
ResourceName	string	256	No	The name of the service.
Resource Description	string	256	Yes	A description of the service as entered in the OVBPI Modeler.

Table 69 Resources

Column Name	Data Type	Length	Allow Nulls	Description
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.
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: • `Active` means the service is deployed. • `Deleted` means the service is undeployed, but is still used by a flow.

1. 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 70 Node2Resources

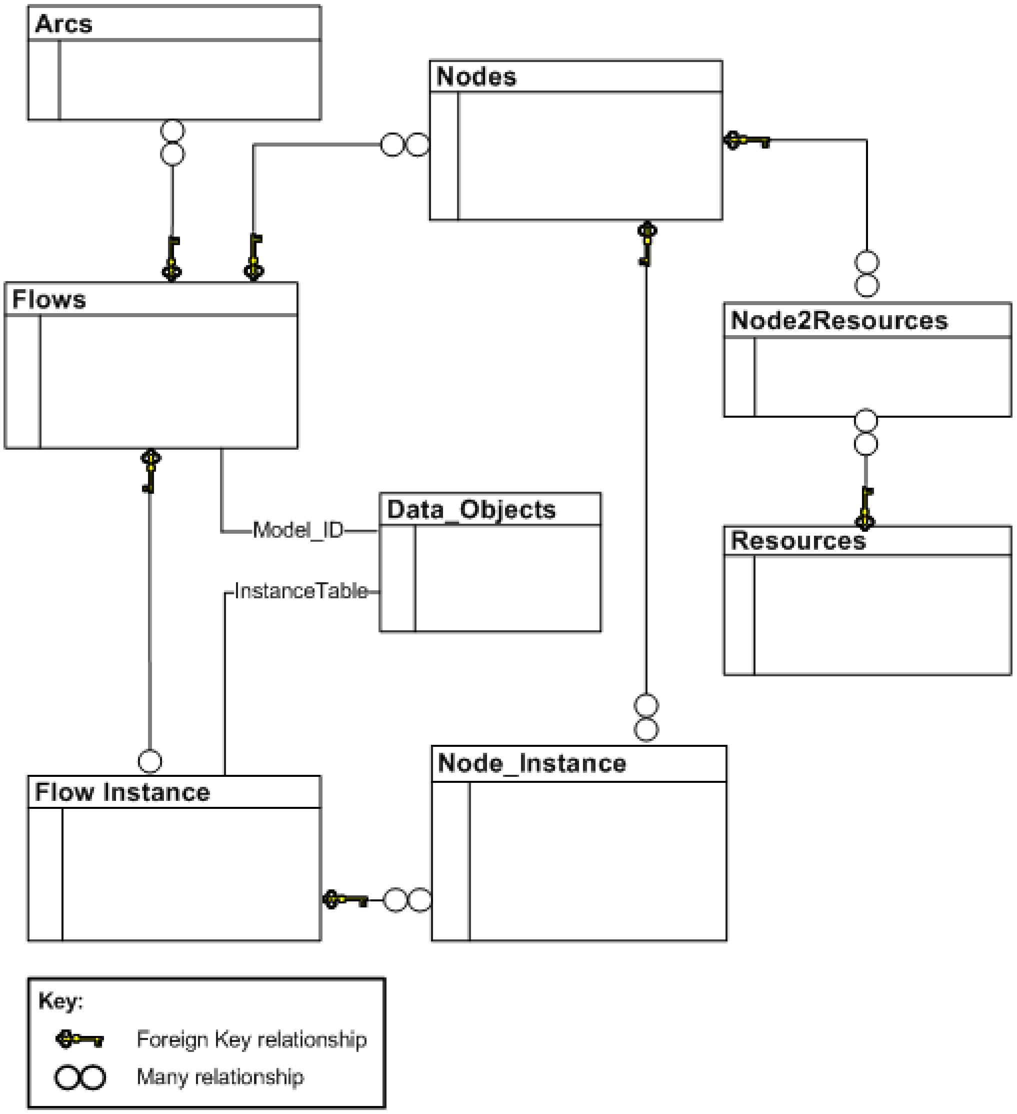
Column Name	Data Type	Length	Allow Nulls	Description
Node_id	string	36	No	Foreign Key[1] to table Nodes (`Node_ID`).
Resource_ID	string	260	No	Foreign Key[1] table Services (`Resource_ID`).

1. 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 285 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 71 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 71 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 72 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 72 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[1].
Model_ID	string	36	Yes	Foreign Key[2]to Data_Objects table (Model_ID).

1. This is the column that uniquely identifies rows.
2. 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 openadapter patients that are stored in the event hospital; the Event Hospital.

The tables structures are defined by openadapter and in this context, a patient is an OVBPI event, which is in an openadapter 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 73 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 73 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 74 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: • `New`, a newly added patient • `Treated`, this patient message has been updated • `Examined`, this patient message has been examined • `Discharge_Wait`, this patient is waiting to be discharged • `Dishcharge`, this patient has been discharged • `Re_News`, this patient has been re-added to the hospital • `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 74 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 74 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 75 on page 343 lists all the OVBPI database tables and Table 76 on page 346 lists all the views.

Table 75 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 75 Complete List of OVBPI Database Tables

Table Name	Internal Use Only (yes/no)
FlowNodeZones	yes
Flows	no
Flow_Instance	no
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_Staging_Statistics	yes
NodeIdAndResourceStatus	yes
NodeInstToBeDeleted	yes
Nodes	no
Nodes2Resources	no
Node_Instance	no

Table 75 Complete List of OVBPI Database Tables

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

The following table lists all the OVBPI database views.

Table 76 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 77 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
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

Servlet Container Authentication

This appendix describes how to modify the Memory Realm mechanism within Tomcat to modify the login details of the following OVBPI components:

- Repository Explorer (and Modeler)
- Intervention Client
- Notification Server Administration Console
- Metric Definer
- Event Injector

  Note that the Event Injector is one of the OVBPI contributed components and is not installed as part of OVBPI, you need to manually copy it from the OVBPI distribution media.

Each of these components requires you to provide a username and password in order to access the Web screens. Following a new OVBPI installation, the username and passwords are predefined as:

- Username: `admin`
- Password: `ovbpi`

Tomcat Realm Mechanism

OVBPI uses Tomcat as its Servlet Engine to present JSPs to the user, through a Web browser; starting the Servlet Engine is described in section Starting and Stopping the OVBPI Server Components on page 66. For authentication, OVBPI uses the Tomcat Realm mechanism, specifically it uses the Memory Realm mechanism, which is more fully described in the Tomcat documentation at the following URL:

`http://jakarta.apache.org/tomcat/tomcat-5.0-doc/realm-howto.html`

Within the Tomcat documentation, a Realm is described as a database of usernames and passwords, which identify valid users of a web application (or set of web applications), plus a list of roles associated with each valid user.

Within OVBPI the following roles have been defined:

- `ovbpi-model-repository`, which is the role for the Repository Explorer and the Modeler; see Chapter 6, OVBPI Modeler Administration and Chapter 8, Repository Explorer.
- `ovbpi-notify-admin`, which is the role for the Notification Server Web Administration Console; see Chapter 9, Notification Server Configuration.
- `ovbpi-int-client`, which is the role for the Intervention Client; see Chapter 10, Intervention.
- `ovbpi-metric-definer`, which is the role for the Web-based Metric definition interface; see Chapter 7, OVBPI Metric Definer Administration.
- `ovbpi-event-injector`, which is for a contributed tool that is provided on the OVBPI media.

There is an XML file called `tomcat-users.xml` that defines the credentials for these roles. The file is located at:

```
OVBPI-install-dir/nonOV/jakarta-tomcat-5.0.19/conf/
tomcat-users.xml
```

Following a new installation, the file has the following content:

```
<?xml version='1.0' encoding='utf-8'?>
<tomcat-users>
  <role rolename="ovbpi-model-repository"/>
  <role rolename="ovbpi-int-client"/>
  <role rolename="ovbpi-notify-admin"/>
  <role rolename="ovbpi-metric-definer"/>
  <role rolename="ovbpi-event-injector"/>
  <user username="admin" password="ovbpi"
roles="ovbpi-notify-admin,ovbpi-int-client,ovbpi-event-injector,
ovbpi-metric-definer,ovbpi-model-repository"/>
</tomcat-users>
```

By default, all these roles use the same credentials. You are strongly advised to assign different credentials to the different roles in order to increase the security of your system. This is particularly the case for the Intervention Client, where you can delete active flow instances and data instances.

Modifying Login Credentials

To assign different credentials, you need to edit the XML file that defines the roles and the login credentials. The following is an example of how you can do this:

```
<?xml version='1.0' encoding='utf-8'?>
<tomcat-users>
  <role rolename="ovbpi-model-repository"/>
  <role rolename="ovbpi-int-client"/>
  <role rolename="ovbpi-notify-admin"/>
  <role rolename="ovbpi-metric-definer"/>
  <role rolename="ovbpi-event-injector"/>
  <user username="model-repository" password="admin"
    roles="ovbpi-model-repository"/>
  <user username="notif-admin" password="admin"
    roles="ovbpi-notify-admin"/>
  <user username="interv" password="secret"
    roles="ovbpi-int-client"/>
  <user username="metric" password="secret"
    roles="ovbpi-metric-definer"/>
  <user username="inject" password="inject"
    roles="ovbpi-event-injector"/>
</tomcat-users>
```

In this example, each of the clients now has a separate username and password:

- The `ovbpi-model-repository` role for the Model Repository (used within the OVBPI Modeler) has a username `model-repository` with a password of `admin`.
- The `ovbpi-notify-admin` role for the Notification Server Web Administration Console has a username `notif-admin` with a password of `admin`.
- The `ovbpi-int-client` role for the Intervention Client has a username of `interv` and a password of `secret`.
- The `ovbpi-metric-definer` role for the Metric definer interface has a username of `metric` and a password of `secret`.
- The `ovbpi-event-injector` role for the contributed tool has a username and password of `inject`.

To make changes to the `tomcat-users.xml` file, complete the following steps:

1. Make sure that you have logged out of the OVBPI Modeler, Notification Server Administration Console and the Intervention Client.
2. If you are using the Event Injector contributed component, you must also close the Event Injector.
3. Open `tomcat-users.xml` in a text editor:

   ```
   ovbpi-install-dir/nonOV/jakarta-tomcat-5.0.19/conf/
   tomcat-users.xml
   ```
4. Make your required changes to the file.
5. Save and close the `tomcat-users.xml` file.
6. Stop and restart the `Servlet Engine` from the `Status` pane of the OVBPI Administration Console for the changes to take effect.

Memory Realm is a simple implementation of the Tomcat 5 Realm interface. It is described in the Tomcat documents as a demonstration implementation and not designed for production use. You should use an alternative Tomcat authentication implementation if you want to ensure the security of your system. Refer to the Tomcat documentation for details of how to do this.

You are also advised to set session time outs for the browsers sessions. This minimizes unauthorized access when the OVBPI Web interfaces are running.

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 78 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 Engine log file: `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 Engine log file: `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 79

Operator	**Behavior**
Arithmetic: • `+` • `-` • `*` • `/` • `%`	Supported between Numeric properties or values that can be coerced to numeric using the assignment rules described in Table 78.
Comparison: • `<` • `<=` • `=` • `!=` • `>` • `>=`	The comparison operators can be used to provide a: • numeric comparison between Numeric properties or values that can be coerced to numeric using the assignment rules described in Table 78. • date comparison between Date properties. • alphabetic comparison (locale-specific) between String properties or values that can be coerced to Strings using the using the assignment rules described in Table 78 (including Boolean). The result of a comparison is always a boolean value.
String tests: • `starts()` • `ends()` • `contains()` • `in()`	Performs a test on a String property. Note that these tests cannot be applied to non-string properties.

Table 79

Operator	**Behavior**
Logical: • `!` (Not) • `&&` (And) • `|| (Or)`	Supported between Boolean properties or values that can be coerced to boolean using the assignment rules described in Table 78.
Conditional value `if, then, else`: • `?:`	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 80

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

D

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 362 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 *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 366.

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 366. 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 Concepts and Modeling Flows* and the *OpenView Business Process Insight Integration Training Guide - Modeling Flows*.

The methods used for the progression rule styles are listed in Table 81 on page 364.

Table 81 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 81 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 82 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. 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 82 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. 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. 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.` 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”`

index