HP OpenView
Business Process Insight

Integration Training Guide
Monitoring Service Desk

Software Version: 02.00

January 2006

Legal Notices

Warranty

Hewlett-Packard makes no warranty of any kind with regard to this document, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Hewlett-Packard shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

A copy of the specific warranty terms applicable to your Hewlett-Packard product can be obtained from your local Sales and Service Office.

Restricted Rights Legend

Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause in DFARS 252.227-7013.

Hewlett-Packard Company
United States of America

Rights for non-DOD U.S. Government Departments and Agencies are as set forth in FAR 52.227-19(c)(1,2).

Copyright Notices

Trademark Notices

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

Microsoft® is a US registered trademark of Microsoft Corporation.

UNIX® is a registered trademark of The Open Group.

Windows® and MS Windows® are US registered trademarks of Microsoft Corporation.

All other product names are the property of their respective trademark or service mark holders and are hereby acknowledged.

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 training guide looks at the OpenView Service Desk Process Insight module, which provides template flows and adaptors to help you to monitor OpenView Service Desk (OVSD) using OpenView Business Process Insight (OVBPI).

This chapter provides an overview of what makes up the OVSD Process Insight module.

Prerequisites

To configure and/or extend the OVSD Process Insight module you need to be comfortable modeling OVBPI flows, configuring adaptors, configuring SQL triggers, and working with OVBPI dashboards.

This training guide assumes that you have read the following guides:

- *OVBPI Integration Training Guide - Modeling Flow*
- *OVBPI Integration Training Guide - Business Events*
- *OVBPI Integration Training Guide - Customizing Dashboards*

OVSD Process Insight Components

With the OVSD Process Insight module, you are able to visualize your ITIL processes within OVBPI, and monitor how they are performing.

The OVSD Process Insight module consists of the following:

- A set of ITIL flow definitions:

  ```
  Service Calls
  Incidents
  Problems
  Changes
  Work Orders
  ```

- A set of adaptors

  One adaptor for each ITIL flow.

- A set of OVSD database triggers.
- A customized Web dashboard.

  You can use the standard OVBPI dashboard, however, a customized dashboard is provided to give an example of how you can tailor a dashboard specifically for OVSD to focus on your ITIL processes.

Integration Templates

The OVSD Process Insight module provides five ITIL flow definitions and the associated adaptors, SQL triggers and customized Web dashboard. But not every OVSD installation is going have the same ITIL flow definitions.

The OVSD Process Insight module is provided as a set of integration templates. It is fully expected that OVBPI Consultants will be engaged to enhance and tailor the integration to match your OVSD installation.

How to customize the OVSD Process Insight module is covered in Chapter 5, Advanced Customization.

Supported OVSD Product Version

The OVSD Process Insight module has been tested against **OVSD 4.5** running **Service Pack 13**.

The OVSD Process Insight module requires that the OVSD Database Reporting Views exist. (Refer to OVSD Views on page 24 for further details.)

The Custom Web Dashboard

Although you can use the standard OVBPI Web dashboard to view your ITIL flow instances within OVBPI, the customized Web dashboard provided with the OVSD Process Insight module presents the information in a way that is more focused around the five ITIL flows of the OVSD Process Insight module.

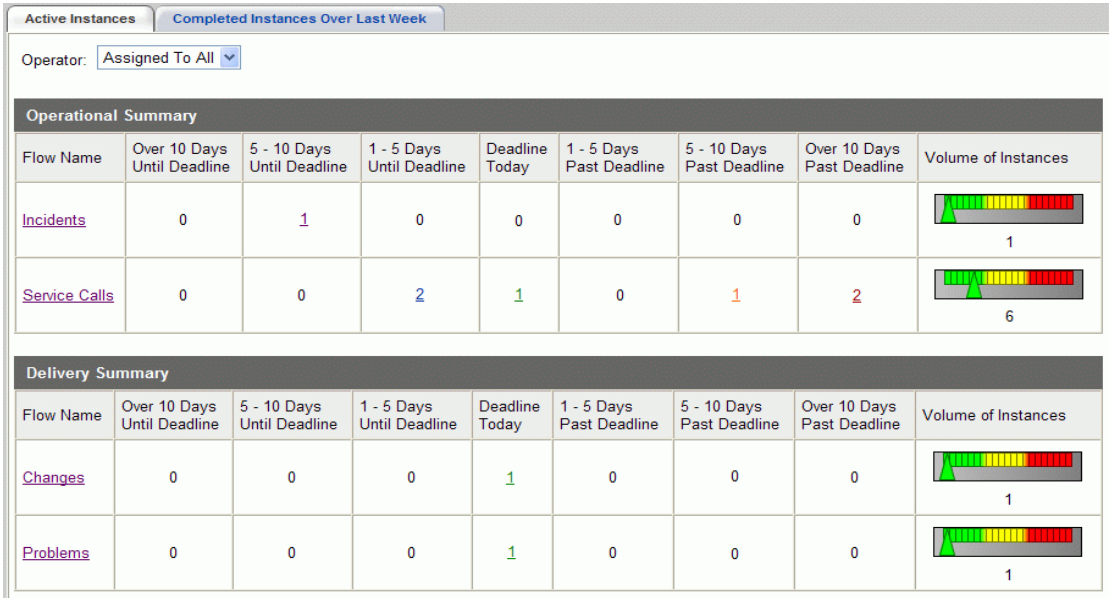
Let's have a look at the main screens of the customized Web dashboard.

Main Page

The custom Web dashboard allows you to see at a glance the overall status of your OVSD items.

The main page displays all items relative to their assigned deadline. For example:

Figure 1 Main Page - Deadline Tracking



where:

- You can see at a glance how many instances are due to be completed today (the `Deadline Today` column).

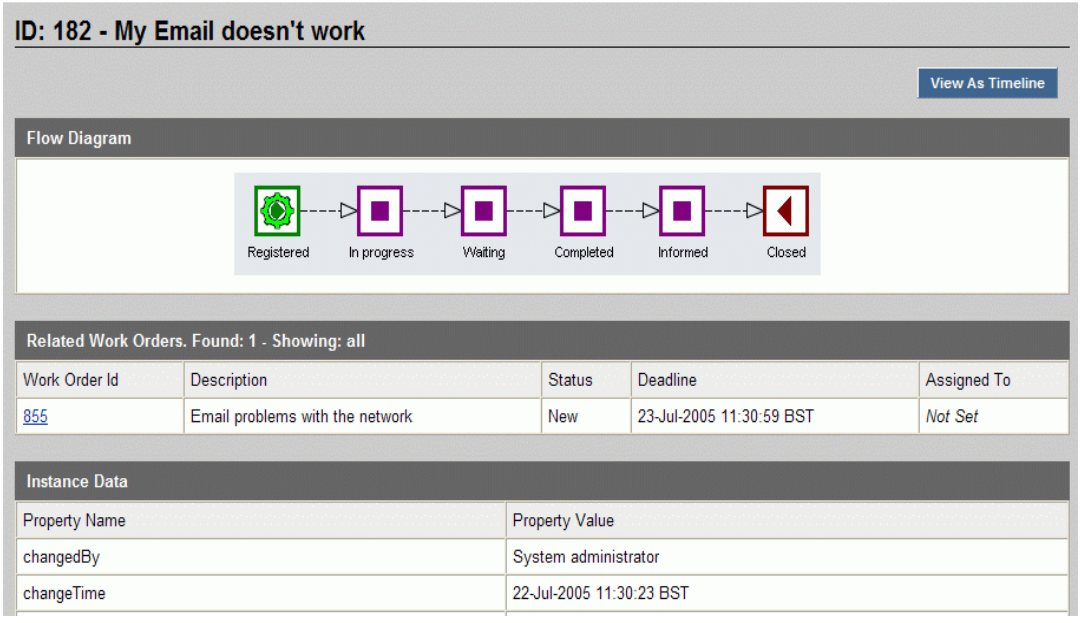
- You can see how many items were due to complete but have overrun by a number of days. You can also see how many items are coming up to their completion deadline.

  For example, the Service Call listing shows that there are 2 calls where the deadline is in 1 to 5 days time, 1 call is due to be fixed today, 1 call is at least 5 days overdue, and 2 calls are at least 10 days overdue.

- You can click on the individual numbers to see just those instances, and then drill into them to see further specific details.
- You can click on the flow names to see the details screens for each flow.
- You can filter the items by operator - to show only the items assigned to a particular operator.
- You can configure the individual column headings, and ranges, to be appropriate for your installation (see Chapter 4, Basic Customization).

The main screen (as shown in Figure 1 on page 13) shows the overall status of four of the five ITIL flows. If you drill into a particular instance you can see how that instance is progressing, and it is at this level that you see any work orders. For example:

Figure 2 Drill Down



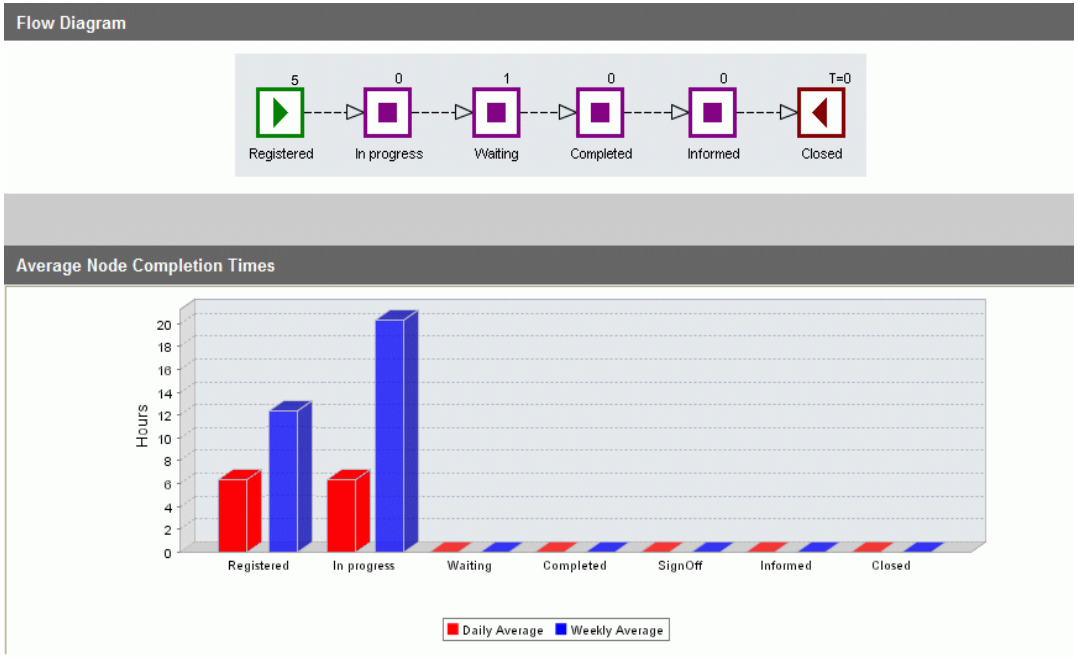
where:

- You see where the item is within its “flow”. It is currently at the `Registered` stage.
- If there are any related work items, they are listed here, and you can drill down into each work order to see how that is progressing.

ITIL Flow Overview

From the main screen, if you click on the actual flow name (for example, `Service Calls`) it takes you to the overview page for that flow. For example:

Figure 3 Overview Page - Service Calls



where:

- Each node in the flow diagram represents a status that a service call can have.
- The numbers above each node in the flow diagram show the number of service calls currently at that step in the process. So you can see how your items are distributed throughout the overall flow.

- The “Average Node Completion Times” bar graph shows the average time that a service call is spending in each of these steps.

  The graph shows the daily average (in red) as well as a rolling weekly average (in blue), allowing you to see how today’s averages compare to the weekly average.

2

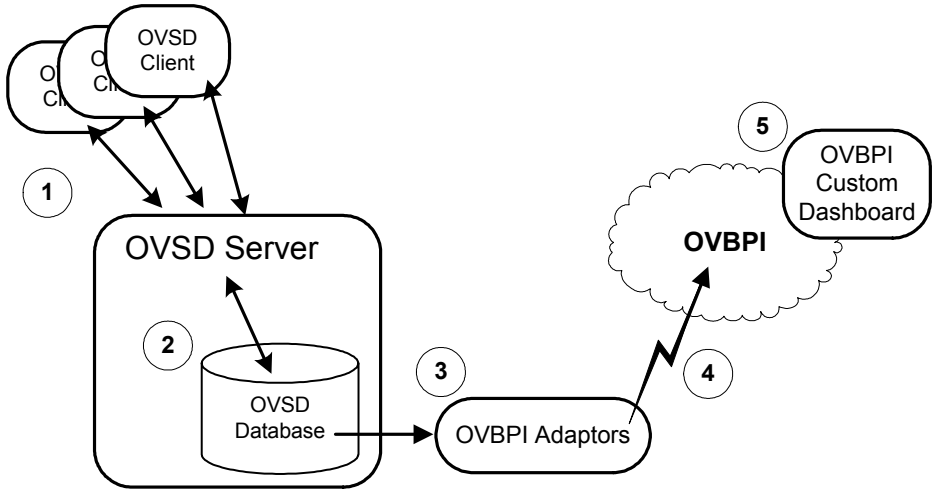
Architecture

This chapter looks at the architecture of the OVSD Process Insight module. This chapter highlights the “touch points” of the OVBPI Integration with OVSD. That is, it looks at where, and how, OVBPI connects into OVSD to be able to monitor the OVSD activity.

The Big Picture

The overall architecture looks as follows:

Figure 4 OVSD Process Insight module - Overview



where:

1. OVSD Clients enter and update Service Calls, Problems, etc. in the normal way.
2. The OVSD Server maintains these updates in the OVSD Database.
3. OVBPI adaptors monitor these updates to the OVSD Database.
4. The OVBPI adaptors send all the updated information to the OVBPI server as business data events.

   OVBPI stores and collates these business events against your ITIL flows, in the normal way.
5. You use the OVBPI custom dashboard, from within a Web browser, to view your ITIL flows and monitor how they are performing.

Server Configurations - Local and Remote

The OVSD Process Insight module does not enforce any strategy as far as where you run OVBPI relative to OVSD. That is, they can both be installed on the same machine or separate machines. The choice is yours.

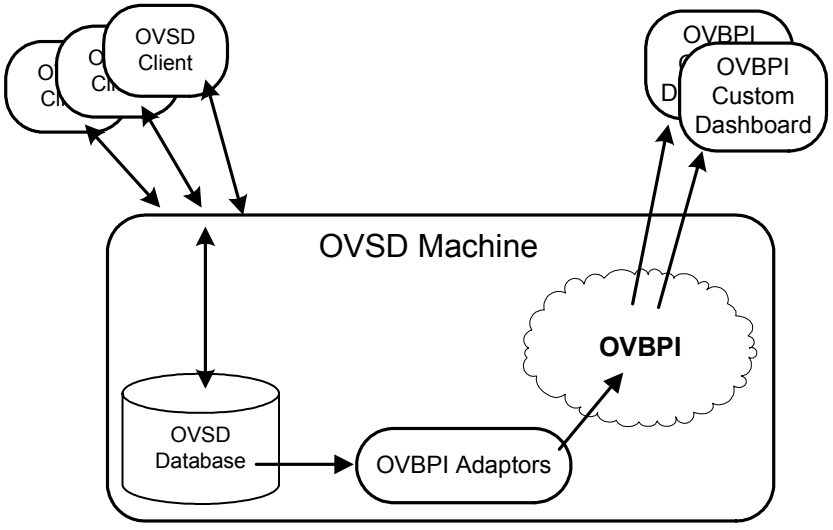
However, for a production system you would probably run OVBPI on a machine separate from the OVSD server.

You then have the choice of where to run the OVBPI adaptors. These can run local to the OVSD database on the OVSD machine, or remotely from the OVSD database on the OVBPI machine.

So you have the following configuration strategies available to you:

On Same Machine

Figure 5 Installation - Same Machine

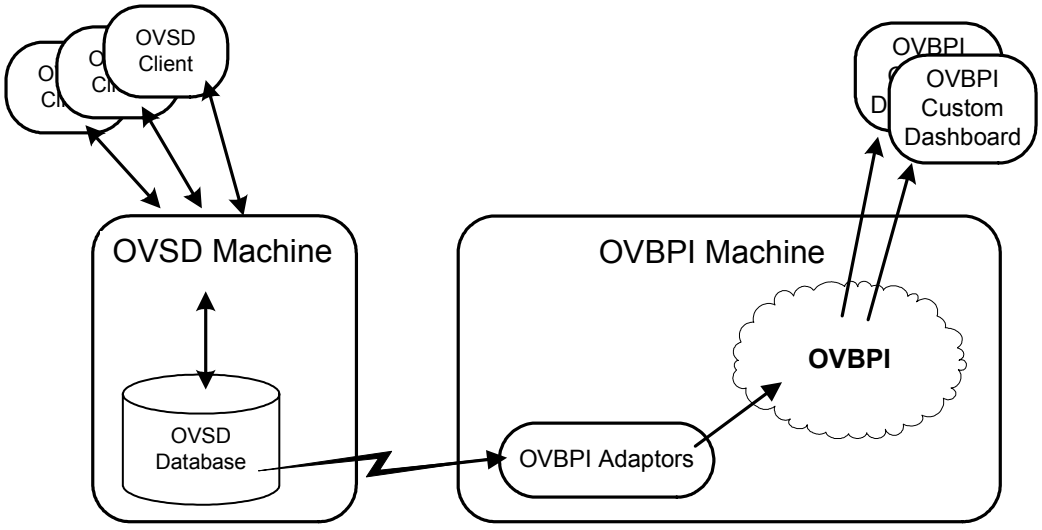


This is most likely to be the configuration you use when giving a demonstration, where you only have a single PC.

This configuration is not ideal for production use.

Separate Machines - Remote Adaptors

Figure 6 Installation - remote adaptors

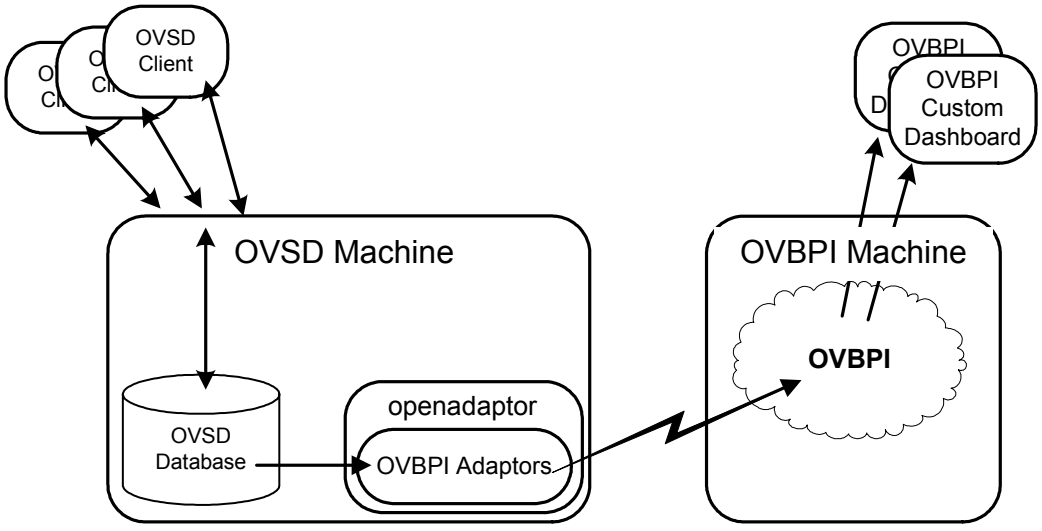


With OVBPI and OVSD on separate machines, you have the choice of where to run the actual OVBPI adaptors. The configuration shown in Figure 6 has the advantage that you do not need to install openadaptor on the OVSD machine. openadaptor in installed as part of the OVBPI installation, so the OVBPI adaptor automatically has the necessary environment to run.

You would need to test the network impact of the OVBPI adaptor issuing its SQL statements across the network to the OVSD database. This would be the deciding factor in choosing this installation architecture.

Separate Machines - Local Adaptors

Figure 7 Installation - local adaptors



If you run the OVBPI adaptors on the OVSD server then you also need to install openadaptor on the OVSD server. Refer to the *OVBPI Integration Training Guide - Business Events* for details of how to install a standalone version of openadaptor.

Obtaining the Data from OVSD

So how do the OVBPI adaptors monitor the OVSD database and extract the data to send to OVBPI? Let's expand step 3 of Figure 4 on page 18.

To provide reliable data feeds with minimal impact on the normal operation of OVSD, the extraction is implemented using SQL Triggers on a small number of OVSD database tables.

SQL Triggers

Consider for a moment how you might track a Service Call. When a Service Call is first created, you want to be told about it. Whenever this Service Call is subsequently modified, you want to be told about it. Within the OVSD database, when a Service Call is created or modified, a new record is written to the ITSM_HISTORYLINES_SERVICECALL data table. Thus, by placing a single trigger on this table, you are able to see each and every Service Call and any modifications.

When a new record is written to the ITSM_HISTORYLINES_SERVICECALL data table, the SQL trigger is executed. The trigger pulls out some key information from the record being inserted into the history line table, and insert this key information as a new record into a special OVBPI data table, called OVBPI_ITEM_CHANGE_HIST.

There are similar history line tables for Incidents, Changes, Problems and Work Orders. And there are triggers for each of these tables. All the triggers write their output to the one table - the OVBPI_ITEM_CHANGE_HIST table.

OVBPI Item Change History Table

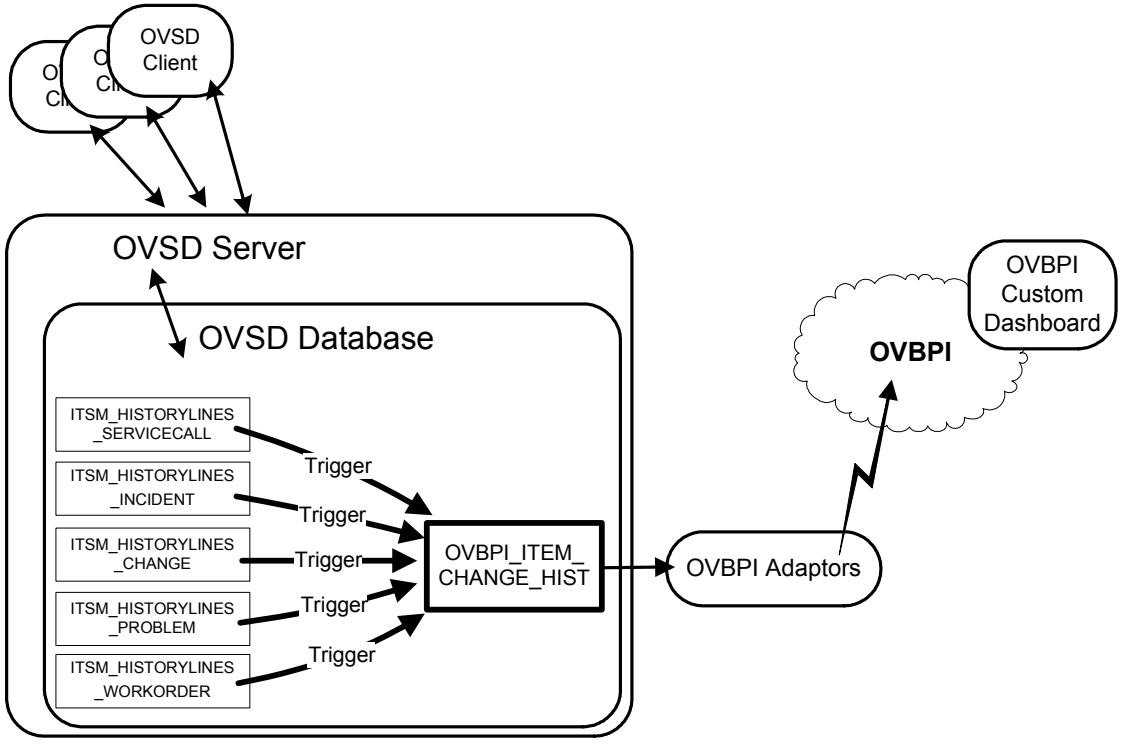
The OVSD Process Insight module installs a new data table called: OVBPI_ITEM_CHANGE_HIST. This table is a staging area (often called a "buffer table") for all the updates that happen to your Service Calls, Incidents, Changes, Problems and Work Orders. The OVBPI_ITEM_CHANGE_HIST buffer table needs to be within the OVSD database.

A set of OVBPI adaptors polls this buffer table, picking up all the updates from OVSD and sending them, as business data events, to OVBPI.

As each record is processed from this buffer table, the record is removed - so the OVBPI_ITEM_CHANGE_HIST should not grow without bound.

So the overall architecture, showing the SQL triggers, looks like Figure 8:

Figure 8 OVSD Process Insight - SQL Triggers



where:

- The five SQL triggers and the OVBPI_ITEM_CHANGE_HIST buffer table are necessary to support the integration with OVBPI.
- As new Service Calls, Incidents, etc. come in, or are modified, the SQL triggers write key information into the OVBPI_ITEM_CHANGE_HIST buffer table.
- The OVBPI adaptors process the records in this OVBPI_ITEM_CHANGE_HIST buffer table, and send the details to OVBPI for monitoring purposes.

OVSD Views

The data records written into the `OVBPI_ITEM_CHANGE_HIST` buffer table are fairly small. Each record is basically just the internal OVSD Object ID (OID) of the item that has been modified, a timestamp of when this action occurred, and what type of modification this is. The execution of the SQL trigger needs to be efficient so as not to adversely affect the normal day-to-day running of the OVSD system.

This means that, as each OVBPI adaptor processes these records from the `OVBPI_ITEM_CHANGE_HIST` buffer table, the adaptor needs to enrich the data by using this OID to obtain additional information about each particular record. This additional data is obtained by accessing the OVSD Reporting Views.

The OVSD Reporting Views are an optional feature within OVSD, but for the OVSD Process Insight module these views **must** exist.

Creating the OVSD Views

The OVSD reporting views are created from within the OVSD Administrator Console.

To start the OVSD Administrator Console:

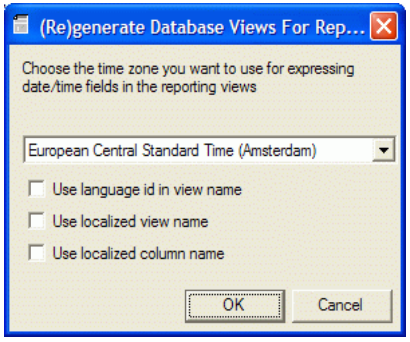
- Log in to the OVSD Client as an OVSD user with administrator capabilities.
- Then select: `Tools->System...`

Once in the OVSD Administrator Console:

- Click on `System Panel` - in the left-hand pane.
- Double-click on `Report Settings` - in the right hand pane.
- On the `General` TAB, click on the `(Re)generate database views for reporting...` button (in the lower part of the dialog)

  This then gives you a dialog box as follows:

Figure 9 OVSD Database Views Creation



This is for an OVSD Service Pack 13 installation - hence you have the three checkboxes for selecting additional language and localization settings.

You then select your time zone, and any options for localized names, and click OK.

Refer to OVSD DB View and Column names on page 33 for more details about recommendations for which localization settings to use, and the impact that selecting these options can have on the OVSD Process Insight module.

List of Views Required

Of the views created when you generate them from within the OVSD Administrator Console, the following is the list of views used by the OVBPI adaptors:

- v_historylineincident
- v_incident
- v_historylineservicecall
- v_servicecall
- v_historylinechange
- v_change
- v_historylineworkorder
- v_workorder
- v_historylineproblem
- v_problem
- v_person
- v_priority

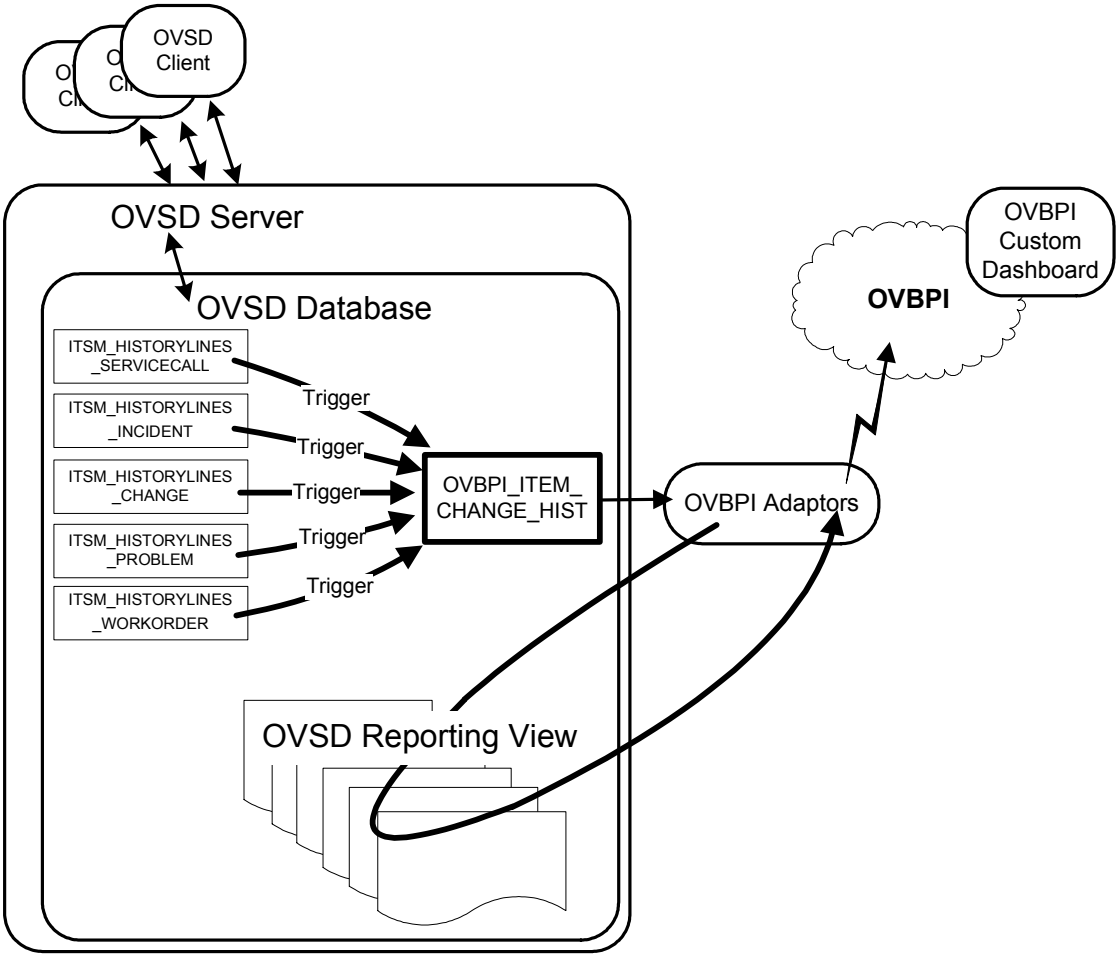
When you generate the views (from within the OVSD Administrator Console), the localization options you choose can affect the names of these views listed above. This is all configurable when installing the OVBPI adaptors - see The Adaptors on page 31 for more details.

Detailed Architecture Diagram

The overall architecture of the OVSD Process Insight module can be detailed further by showing the use of the OVSD views.

The overall architecture diagram now looks like Figure 10:

Figure 10 OVSD Process Insight - in Detail



Summary of OVSD “Touch Points”

Here is a summary of the main “touch points” of the OVSD Process Insight module. This details the additions and modifications the OVSD Process Insight module makes to the OVSD database at configuration time, and the OVSD database requirements at run time.

Configuration

When the OVSD Process Insight module is configured, the following additions are made to your OVSD database:

- A new data table is created, called: `OVBPI_ITEM_CHANGE_HIST`
- SQL Triggers are installed on the following OVSD data tables:

```
ITSM_HISTORYLINES_SERVICECALL
ITSM_HISTORYLINES_INCIDENT
ITSM_HISTORYLINES_CHANGE
ITSM_HISTORYLINES_PROBLEM
ITSM_HISTORYLINES_WORKORDER
```

  As OVSD inserts records into these history line tables, these triggers insert corresponding records into the `OVBPI_ITEM_CHANGE_HIST` table.

  All triggers write to the same `OVBPI_ITEM_CHANGE_HIST` table.

Run Time

At run time, the OVBPI adaptors process the records from the `OVBPI_ITEM_CHANGE_HIST` table, and the OVBPI adaptors use the OVSD Database Reporting Views to gain additional information for each record.

So, the OVSD Administrator is required to set up the OVSD Database Reporting Views before you can configure and run the OVSD Process Insight module.

3

The OVSD Process Insight Module

This chapter explains how to configure and deploy the OVSD Process Insight module.

Module Files

The files that make up the OVSD Process Insight module are all located under your OVBPI installation directory:

- *OVBPI-Install_dir*\examples\bia\ServiceDeskFlows

  This directory holds the ITIL flow definitions. These are ZIP files ready to be imported directly into the OVBPI Modeler.

- *OVBPI-Install_dir*\examples\bia\ServiceDeskAdaptors

  This directory contains everything to do with your adaptors. It includes things such as adaptor property files, adaptor start and stop scripts, and SQL trigger definition scripts.

- *OVBPI-Install_dir*\nonOV\jakarta-tomcat-5.0.19\webapps\ovbpi_sd_dashboard.war

  This is the OVSD Process Insight custom dashboard. This WAR file is already expanded for you (under the webapps\ovbpi_sd_dashboard directory) and ready for use.

The next steps are to deploy each flow, configure the adaptors and use the custom dashboard.

Deploy The Flows

Deploying the flows is straightforward. You carry out the following steps on your OVBPI server:

- Start up the OVBPI components.
- Run the OVBPI Modeler.
- For each flow in the directory:

  OVBPI-Install_dir\examples\bia\ServiceDeskFlows

  — Import the flow definition.

  — Deploy the flow definition.

The Adaptors

It would be great if all the adaptors were configured and ready to run. However, because OVSD is quite configurable, there are a number of key items that you need to provide before the adaptors can work for your installation.

SD_Adaptor_Config.properties

All the key configuration items that can be specific to your OVSD installation are set up as properties within the file:

OVBPI-Install_dir\examples\bia\ServiceDeskAdaptors\
SD_Adaptor_Config.properties.

You must edit the `SD_Adaptor_Config.properties` file, and go through **every** property, making sure that they are correct for your OVSD installation.

When you first look at the file you may be surprised at the large number of configurable properties. Let's have a look at them.

The properties basically fall into the following categories:

1. OVSD DB View and Column names

   These make up the bulk of the file. These are the names of each of the OVSD database views that the adaptors need to access, and the column names within these OVSD views. (See OVSD DB View and Column names on page 33 for more details. There is a way to set up OVSD such that the bulk of the database column names match.)

2. OVBPI host name details

3. OVSD Database connection details

   These are details such as the name of the OVSD Database, the JDBC driver, JDBC connection URL details, the user name and password.

   There are two blocks of OVSD database properties in the configuration file - one for MSSQL and one for Oracle. You comment in/out the necessary lines.

   Note that the OVSD database password needs to be specified in encoded form. To produce an encoded password, run the command:

   ```
   $ java -classpath OVBPI-install-dir\java\bia-event.jar
                  org.openadaptor.adaptor.util.Encoder password
   ```

   (all on one command line)

   where:

   — You supply the full path for the `bia-event.jar` file

   — You supply your password in plain text

   — The encoded version of the password is output to the command window display

4. Adaptor Remote Control Ports

   These values just need to be numbers of ports available on your system.

5. Adaptor Commit SQL command

   This property lets you set the behavior for the adaptors once they have processed records from the `OVBPI_ITEM_CHANGE_HIST` table.

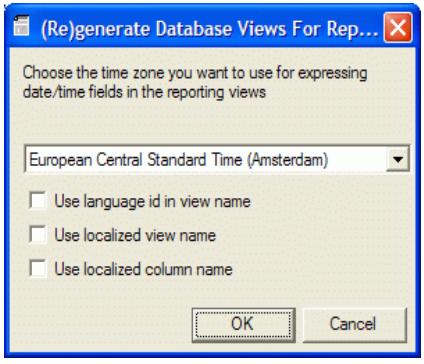
If you do not set all properties in `SD_Adaptor_Config.properties` correctly for your installation, you may not realize that there is a problem until you actually have everything up and running. See Run Time Adaptor Errors on page 75 for an example of errors that might occur.

It is essential to take your time setting the `SD_Adaptor_Config.properties` file and get all the values correct for your installation.

OVSD DB View and Column names

When you generate the Database Reporting Views from within the OVSD Administrator Console, you can select various options regarding localization. These options determine the actual names used to create the views and the columns within the views.

Figure 11 OVSD Database Views Creation



This is for an OVSD Service Pack 13 installation - hence you have the three checkboxes for selecting additional language and localization settings.

If you leave all three checkboxes empty (as shown in Figure 11), then the view names, and their column names, that are generated should match those provided by default in the `SD_Adaptor_Config.properties` file.

You need to go through and check every property, and ensure they are correct for your OVSD database.

For example:

The default `SD_Adaptor_Config.properties` file has the line entry:

```
INCIDENT_VIEW           = v_incident
```

This is saying that the property `INCIDENT_VIEW` needs to be set to the name of the incident view in your OVSD database. On your installation it might be named `v1033_incident`, or it could be something else. You need to find the name for the incident view in your database and set the property in the `SD_Adaptor_Config.properties` file to that name.

Generate the Adaptors and Triggers

Once you have set up the `SD_Adaptor_Config.properties` file, you are ready to generate the adaptor configuration files and the SQL trigger scripts.

You run the script `sd_createadaptors.bat` on the OVBPI machine:

- Open a Console window on the OVBPI machine.
- Within this console window, set the two environment variables:
  - `OVBPI_ROOT`

    Set this to the installation directory for OVBPI on your machine.

    The default is: `C:\Program Files\HP Openview\OVBPI`
  - `JAVA_HOME`

    Set this to the installation directory for Java on your machine.

    The default is: `C:\java`
- Then run the script: `sd_createadaptors.bat`

  The script reads the template adaptor configuration files, and SQL trigger files (found under the `template` subdirectory), and substitutes the property values as specified in your `SD_Adaptor_Config.properties` file.

The `sd_createadaptors` script creates the following files in the *`OVBPI-ROOT`*`\examples\bia\ServiceDeskAdaptors` directory:

- The adaptor configuration files:

```
SD_Poll_ServiceCall.props
SD_Poll_Incident.props
SD_Poll_Change.props
SD_Poll_Problem.props
SD_Poll_WorkOrder.props
```

- The trigger files:

```
CreateServiceDeskTriggers.sql
DropServiceDeskTriggers.sql
```

- The stop adaptor scripts:

```
sd_stopadaptors.bat
sd_stopadaptors.sh
```

Regenerating the Adaptors and Triggers

If you need to make alterations to any of the properties in your `SD_Adaptor_Config.properties` file, simply re-run the `sd_createadaptors` script, and the changes are reflected throughout all the regenerated files.

Install the Triggers

The SQL triggers need to be installed on the OVSD server's database.

After running the `sd_createadaptors.bat` file on the OVBPI machine, you have two new SQL files in the `examples\bia\ServiceDeskAdaptors` directory:

- `CreateServiceDeskTriggers.sql`

  The script to create the necessary triggers for the OVSD Process Insight integration.

- `DropServiceDeskTriggers.sql`

  A script for removing the OVSD Process Insight triggers.

To install the triggers you need to execute the `CreateServiceDeskTriggers.sql` command file whilst logged on to the OVSD Database as the database administrative user.

The `CreateServiceDeskTriggers.sql` script can be run from within your favorite SQL utility.

For example: (for MSSQL)

```
$ isql -U servicedesk -P sd -i CreateServiceDeskTriggers.sql
```

where:

- The user name `ServiceDesk` is the Service Desk administrative user.
- The password in this example is `sd`.

Verify OVSD with the Triggers

Having installed the OVSD Process Insight triggers into the OVSD database, it is important to verify that the triggers work and that the SD Client is still able to create and/or modify items such as Service Calls, Incidents, Changes, Problems and Work Orders.

Enter Some Dummy Data

In a development installation you could simply create a new item of each type - Service Call, Incident, Change, Problem and Work Order.

If the triggers are installed correctly there should be no database errors when creating any of these new items.

After creating each new item you should see new data records written to the `OVBPI_ITEM_CHANGE_HIST` table. You also see one or more records written to this OVBPI table for each new item you create.

If you are able to create new items within your SD Client and you see data being written to the `OVBPI_ITEM_CHANGE_HIST` table, then the triggers are installed correctly and your OVSD installation is functioning correctly.

If you are working on a production OVSD system you probably do not wish to create “dummy” items. That’s fine. As Calls/Incidents/etc. come in, your triggers are actioned and this verifies that everything is working.

Removing Process Insight Triggers

If you encounter problems and you are unable to create items from within your SD Client, you need to drop all the OVSD Process Insight triggers whilst you try and work out what is wrong with them.

To help you drop the triggers you can use the script: `DropServiceDeskTriggers.sql`.

Start the Adaptors

Now that the triggers are in place and working, you are able to start up the adaptors. If you are running the adaptors on an OVBPI Machine then you can go straight to step 3. If you are not running the adaptors on the OVBPI machine then start at step 1.

1. Install openadaptor

   If you are running the adaptors an a server that does not have OVBPI installed, then you need to install the openadaptor software.

   Refer to the *OVBPI Integration Training Guide - Business Events* for details on how to install a standalone version of openadaptor.

2. Copy the adaptor files

   You need to copy the following files from the *OVBPI-Install-Dir*`\examples\bia\ServiceDeskAdaptors` directory on your OVBPI machine, to any directory you choose on your non-OVBPI machine:

   — The adaptor configuration files:

   ```
   SD_Poll_ServiceCall.props
   SD_Poll_Incident.props
   SD_Poll_Change.props
   SD_Poll_Problem.props
   SD_Poll_WorkOrder.props
   ```

   — The start and stop adaptor scripts:

   ```
   sd_startadaptors.bat(.sh)
   sd_startadaptor.bat(.sh)
   sd_stopadaptors.bat(.sh)
   ```

   You choose the `.bat` or `.sh` files depending on your operating system.

3. Set the openadaptor Classpath

   If you have just installed a standalone openadaptor then you have already carried out this step. If you are running on an OVBPI server then you need to make sure that your openadaptor installation is configured correctly.

   There are script files supplied with openadaptor that may or may not be auto-configured at installation time. You need to check these and make sure that the classpath has been set correctly:

— change directory to the *OA-install-dir*\examples directory.

For an OVBPI installation, *OA-install-dir* is going to be *OVBPI-install-dir*\nonOV\openadaptor\1_6_5

— Edit the appropriate set_classpath_jdk_* file for your installation.

— Make sure that the CLASSES_DIR= variable is set to the fully qualified path of the classes directory for your openadaptor installation...and **not** ..\classes.

4. Run the Adaptors

To run the adaptors:

— Open a Console window, and change directory to where you have your adaptor files.

On an OVBPI server this is the directory:

OVBPI-Install-Dir\examples\bia\ServiceDeskAdaptors

— Within this console window, set the environment variables:

– OVBPI_CLASSES

Set this to the directory containing the bia-event.jar file.

The default is: C:\Program Files\HP Openview\OVBPI\java

– OA_ROOT

Set this to the installation directory for openadaptor on your machine.

The default is: C:\Program Files\HP Openview\OVBPI\nonOV\openadaptor\1_6_5

– JAVA_HOME

Set this to the installation directory for Java on your machine.

The default is: C:\java

— Run the script: sd_startadaptors.bat (.sh)

This starts all five adaptors.

Start a Single Adaptor

If you need to start a single adaptor, then you can use the script: `sd_startadaptor.bat(.sh)`.

For example, to start up the Service Call adaptor:

```
$ sd_startadaptor.bat SD_Poll_ServiceCall.props ServiceCallAdaptor
```

Stop All Adaptors

The script `sd_stopadaptors.bat(.sh)` can be used to stop all five adaptors.

Just like the `sd_startadaptors` script, the `sd_stopadaptors` script uses the three environment variables:

```
OVBPI_CLASSES
JAVA_HOME
OA_ROOT
```

The Custom Dashboard

With the triggers installed and the adaptors running, you can run the standard OVBPI Dashboard and you are able to see your five ITIL flows.

Indeed, if you have entered any new Service Calls (as suggested in the verification step Verify OVSD with the Triggers on page 37) then you should be able to see them listed in the Dashboard.

Like any OVBPI implementation, the default Dashboard helps you to test that things are working and that changes in OVSD are indeed making it over to OVBPI. However, the default Dashboard is just that, a default dashboard. You want to use the custom dashboard provided as part of the OVSD Process Insight module. The custom dashboard presents a tailored view of the five ITIL flows.

What The Dashboard Displays?

The adaptors monitor the following changes:

- Status changes.
- Deadline changes.
- Assigned Person changes.
- Priority changes.

As these types of changes occur in OVSD, you should be able to see them reflected in the OVBPI ITIL flows within your dashboard...**however**...a flow instance does not occur until an item has changed its state! That is, an OVBPI flow instance is not created for an item until a state change has occurred. Once an item **has changed state,** a flow instance is created and OVBPI then continues to monitor all changes for this item.

So, if you are testing out that changes in the SD Client do indeed cause changes to appear in the OVBPI Dashboard, don't forget that until an item has changed its state (for example, changed from `Registered` to `In progress` - for a service call), it will not appear in the OVBPI dashboard. It will not appear in either the default dashboard or the OVSD Process Insight custom dashboard.

Run the Custom Dashboard

The custom dashboard is already installed under the `webapps` directory and is ready for use...

To run the OVSD Process Insight custom dashboard, point a Web browser at the URL:

`http://`*`hostname`*`:`*`44080`*`/ovbpi_sd_dashboard`

where:

- *`hostname`* is the hostname of your OVBPI machine
- *`44080`* is the default port for the Servlet Engine

For example:

`http://localhost:44080/ovbpi_sd_dashboard`

This works if OVBPI is installed on the same machine as your Web browser.

OVBPI Engine Instance Cleaner

The main page of the custom dashboard allows you to select the tab labelled: `Completed Instances Over Last Week`. However, this `Completed Instances Over Last Week` tab requires that your OVBPI Engine instance cleaner is configured to leave at least one week's (seven days) worth of completed flow instances in the OVBPI Engine database.

For example, if your OVBPI Engine instance cleaner is cleaning out completed flow instances after only 3 days then clicking on the `Completed Instances Over Last Week` tab is only able to present the last 3 days worth of data.

You should set the OVBPI Engine instance cleaner to clean out completed instances after `7` days or more.

Summary of Main Steps

Here is a checklist summarizing the main steps, outlined above, to set up the OVBPI Process Insight module:

- Locate the OVBPI Process Insight files.
- Deploy the ITIL flows.
- Set up the installation specific properties by editing `SD_Adaptor_Config.properties`
- Create the adaptors by running the script `sd_createadaptors.bat`
- Install the triggers into the OVSD database.
- Run the OVBPI Adaptors:
  - Install OA (only if not on an OVBPI server)
  - Copy adaptor files and scripts (only if not on an OVBPI server)
  - Start the adaptors by using the script `sd_startadaptors` (or `sd_startadaptor` for individual adaptors)

  The adaptors require the OVSD Database Reporting Views to exist.
- Use the Custom Dashboard to view your ITIL flows.

4

Basic Customization

The main page of the custom dashboard (the `Service Scorecard` page) has three main sections:

- Operational Summary
- Delivery Summary
- Service Summary

These three sections do allow for some user customization, such as:

- Changing the time interval period (minutes, hours, or days)
- Changing the column headings
- Changing the intervals
- Changing the green/yellow/red ranges for the slider graphics
- Changing the list of services to display in the service summary graphs

It is all done by editing a simple configuration file.

This chapter explains how to make these customizations.

The Configuration Properties File

To modify the custom dashboard settings, you can directly edit the configuration file:

OVBPI-Install-Dir\webapps\ovbpi_sd_dashboard\WEB-INF\
classes\SD_DashboardConfig.properties

...however... any modifications you make are temporary. If someone uses the OVBPI Administration Console to make any changes to OVBPI configuration this removes the changes you have made.

So you have two options available:

1. Make the changes to the `SD_DashboardConfig.properties` file and be aware that they may be lost the next time someone uses the OVBPI Administration Console to reconfigure things
2. Make the changes to the base template files, within the directory `newconfig\DataDir\conf\bia`, and then use the OVBPI Administration Console to apply the changes.

Time Units

You can alter the unit of time used to calculate the deadline lists.

The properties are:

- `OperationSummaryTableUnits=`
- `DeliverySummaryTableUnits=`

These set the unit of time for the appropriate summary section of the page.

Values can be:

```
days
hours
minutes
```

Example

`OperationSummaryTableUnits=hours`

This means that the values given for the operational measurement intervals are read as hour values.

Measurement Intervals

You can alter both the size of each measurement interval and the number of intervals displayed.

The properties are:

- `OperationSummaryTableInterval=`
- `DeliverySummaryTableInterval=`

These set the measurement intervals for the appropriate summary section of the page.

Example 1

```
OperationSummaryTableInterval=-10,-5,-1,0,1,5,10
```

This is the default setting.

The operational section of the page shows calls and incidents divided into these 7 divisions of time. The time units are those set by the `OperationSummaryTableUnits=` property. The intervals listed above are relative to the time the page is displayed.

If `OperationSummaryTableUnits=` is set to days, then these 7 divisions list calls and incidents with deadlines as follows:

```
deadline >= 10 days from now
deadline >= 5 days from now but < 10 days
deadline >= 1 day from now but < 5  days
deadline <  1 day from now, today, or < 1 day in the past
deadline >= 1 day in the past but < 5 days
deadline >= 5 days in the past but < 10 days
deadline >= 10 days in the past
```

Example 2

```
OperationSummaryTableInterval=0,1,5,10
```

The operational section only displays calls and incidents that are due today or overdue.

If `OperationSummaryTableUnits=` is set to days, then these 4 divisions list calls and incidents with deadlines as follows:

```
deadline = today, or < 1 day in the past
deadline >= 1 day in the past but < 5 days
deadline >= 5 days in the past but < 10 days
deadline >= 10 days in the past
```

Example 3

```
OperationSummaryTableInterval=-50,0,20
```

The operational section of the page shows calls and incidents divided into these 3 divisions of time.

If `OperationSummaryTableUnits=` is set to days, then these 3 divisions list calls and incidents with deadlines as follows:

```
deadline >= 50 days from now
deadline <  50 days from now, today, or < 20 day in the past
deadline >= 20 days in the past
```

Column Headings

You can the column headings displayed for each corresponding measurement interval.

The properties are:

- `OperationalColumnHeaders=`
- `DeliveryColumnHeaders=`

These set the column headings for the appropriate summary section of the page.

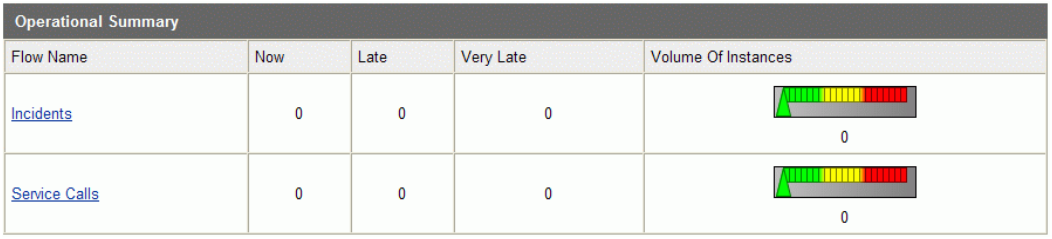
It is up to you to make sure that you column headings match the measurement intervals.

Example

```
OperationSummaryTableInterval=0,1,10
OperationalColumnHeaders=Now,Late,Very Late,Volume Of Instances
```

Would produce an operation section that looks as follows:

Figure 12 Custom Column Headings



Operational Summary

Flow Name	Now	Late	Very Late	Volume Of Instances
Incidents	0	0	0	0
Service Calls	0	0	0	0

Slider Ranges

For each flow, the main page shows a slider graphic. This slider displays the total number of instances that fall within the time intervals you have specified.

For each slider you can specify the min/mid/high/max ranges. That is, you can specify the size of the green/yellow/red areas.

The properties are:

- `OperationalSliderRangesAll=`
  `OperationalSliderRangesOperator=`
- `DeliverySliderRangesAll=`
  `DeliverySliderRangesOperator=`

You enter two sets of ranges. The order of the ranges signals which flow they are for within the appropriate summary section. The flows are listed in alphabetical order on the Web page so you need to specify the ranges for the flows assuming that order of display.

For each summary slider there are two ranges:

- `...RangesAll=`

  Specifies the ranges to be used when the page is displaying for all operators.

- `...RangesOperator=`

  Specifies the ranges to be used when the page is displaying for an individual operator.

Example

```
DeliverySliderRangesAll=0,1000,1000,1050|0,100,200,900
```

Sets the sliders to look as follows:

Figure 13 Custom Slider Ranges

Delivery Summary

Flow Name	Over 10 Days Until Deadline	5 - 10 Days Until Deadline	1 - 5 Days Until Deadline	Deadline Today	1 - 5 Days Past Deadline	5 - 10 Days Past Deadline	Over 10 Days Past Deadline	Volume of Instances
Changes	0	0	0	0	0	0	0	 0
Problems	0	0	0	0	0	0	0	 0

Service List

The main scorecard page of the custom dashboard displays two graphs in the Service Summary section. These graphs list the number of Service Calls and Incidents against a particular service, and the average time to complete these calls and incidents.

You can customize which OVSD services are displayed in these graphs.

The properties are:

- `ActiveTotalServiceNames=`
- `AverageTimeToCompleteServiceNames=`

You provide a comma-separated list of service names. You can chose any service names you require, and the dashboard lists the number and average times of calls and incidents against those services.

5

Advanced Customization

The ITIL flows provided with the OVSD Process Insight module are provided as working examples. It is expected that individual OVSD installations may implement slightly different flows and/or flow states.

This chapter details how to extend the flow definitions to accommodate your individual installation.

Flow Level Changes

The most obvious change that may be required is a change to one or more of the flows. Flow level changes do not require any changes to the OVBPI adaptors or SQL triggers.

Each flow represents the various states that an item can be in during its life cycle, and these are based on the settings of a default OVSD installation. For example, a Service Call goes through the states:

```
Registered
In progress
Waiting
Completed
Informed
Closed
```

and within the flow, there is a node for each of these states.

The flow progression rules tend to be quite straightforward. In the case of the `Service Call` flow, the progression rules tend to be of the form:

```
For NodeA:
Start Condition:  statusValue.after()  == "NodeA"
Stop  Condition:  statusValue.before() == "NodeA"
```

Some flows do vary slightly from this simple one-node-for-each-state format. For example, the `Changes` flow goes through the states:

```
Registered
R && I progress
To be approved
Approved
Implementation
Evaluated
Closed
```

However, the nodes within the flow are:

```
Registered
Risk and Impact Assessment
Approval
Implementation
Evaluation
Closed
```

where:

- The node representing the `R && I progress` state has been renamed to `Risk And Impact Assessment`.

  The engineer developing the flow felt this would be a more appropriate name for the node.

- The two states `To be approved` and `Approved` have been represented within the one node `Approval`.

  Again, this is something that the engineer developing the flow decided to do. The engineer felt that having one node representing approval would make sense.

  The start condition is when the state goes to `To be approved`.

  The end condition is when the state goes to `Approved`.

Representing Additional Nodes/States

Suppose the Service Call states in your OVSD installation include the additional state `SignOff`.

The standard triggers and adaptors that come with the OVSD Process Insight module are able to handle this (and any) additional state, and the state information is sent into the OVBPI Engine. All you need to do is edit the flow definition and add in a new node to represent this additional state.

For example:

You might edit the standard `Service Call` flow to look as follows:

Figure 14 Service Call - Additional Node

where you have added the `SignOff` node after the `Completed` node, because that is where it makes sense for your installation.

You then need to give it progression rules that simply say when the state changes to `SignOff` - you are in the `SignOff` node, and when the state changes from `SignOff` - you are no longer in the `SignOff` node.

For example:

```
Start transition:
   Property: this.data.statusValue
   From:
   To:       "SignOff"

Complete transition:
   Property: this.data.statusValue
   From:     "SignOff"
   To:
```

Renaming Nodes/States

The name of the nodes is totally up to you. You can name a node whatever name you like.

As for the states coming into the flow, the actual states values are used within the progression rules. So if your OVSD installation has the same number of states but the values (names) of these states are different, then you would simply need to open up the flow and edit the progression rules to use the correct state values.

For example:

The default states for a Service Call are:

```
Registered
In progress
Waiting
Completed
Informed
Closed
```

Suppose on your OVSD installation, the states were named:

```
Registered
Work In Progress
Waiting
Completed
Informed
Closed
```

You need to modify the `Service Call` flow and change any progression rules that uses the state value `In progress`, to use the state value `Work In Progress` instead.

You do not have to change the name of the node...but you can if you want to.

Removing Nodes

If there are states in the default flows that you do not use, then you can remove the corresponding nodes from the flows. You should then check all progressions rules to make sure that nothing else is using that state value associated with the node you have just removed.

Event/Data Level Changes

There may be situations where there is additional OVSD data that you would like to monitor within the provided flows. To add additional data to the flows requires significant effort and customization.

There are two types of changes that you might consider:

- Monitoring when something new changes its value

  The current triggers pick up on changes to the items `Status`, `Priority`, `Deadline` and `Assigned Operator`. Suppose you also wanted to be told when (for example) the `Assigned Workgroup` is changes for any items. This would require you modifying the existing database triggers to add an additional test condition. You would then need to modify the adaptors and flow definition(s) to cope with this additional data.

- Bringing in additional data values on existing item changes

  Suppose you wanted to bring in additional data values when (for example) any item changed its `Status`. This would not require any changes to the database triggers. This would just require the adaptors and the flow definition(s) to be modified.

Of course, once this additional data is in the OVBPI system, you need to think about whether you want to make any changes to the custom dashboard. Maybe the data is just additional data with each flow instance. In this case, the current dashboard is able to display that data without any problem. However, maybe the data you are bringing in is something that you would like to highlight within the custom dashboard. In this case, you would need to modify the custom dashboard accordingly.

So, in general, the steps are:

1. Modify the triggers

   This step is only required if you wish the triggers to pick up on changes to the values within an additional data field.

2. Modify the adaptors

   This step is always required. You must edit the adaptor configuration files so that they pick up the new data and send this into OVBPI. This data requires a change to your event definitions - either a new event or extension to an existing event definition.

3. Modify the flow definition (and redeploy)

   For new data to come into OVBPI you need to modify the event definition and the associated data definition (including the event subscriptions).

   You then need to redeploy the flow.

4. Modify the dashboard

   The existing dashboard shows the additional data by default. However, you might like to modify the dashboard to highlight this additional data in some way. For example, you might wish to offer the user the ability to filter their screens based on the new data.

Example: Filtering On Workgroups

The main screen in the custom dashboard for OVSD Process Insight allows you to filter the items to be shown according to the assigned operator.

Suppose that you would prefer to have this filtering based, not on operator, but by assigned workgroup.

How would you go about extending the OVSD Process Insight module to provide for this?

The basic steps are as follows:

1. Extend the SQL in the template database triggers.
2. Extend the template adaptor files.
3. Set up any new tokens required in the `SD_Adaptor_Config.propetries` file.
4. Recreate the adaptor configuration files and trigger files.
5. Remodel the flows:
   - Define the new data property in the associated data definition.
   - Define the new events
   - Define the new event subscriptions
   - Redeploy the flows
6. Install the modified database triggers.
7. Install and run the new adaptors.
8. Customize the Dashboard to allow users to filter by workgroup.

Let's go through and discuss each of these showing code segments and screen shots...

Extend the SQL in the Template Database Triggers

Because you want to receive an event whenever there is a change to the workgroup assignment for an item, you need to extend the SQL triggers to pick up on workgroup changes.

For each trigger, the extension is similar. The only difference is the name of the string you specify in the `ItemType` column.

For example, in the Service Call trigger (`servicecall_status_change`) you simply need to add another `if` statement that tests when the search code value is `To workgroup`. This is the value that is contained in the OVSD history record when this trigger is executed and the change is a workgroup assignment change. Of course, for your OVSD installation this might be different. You would obviously need to check this value for your installation.

The code segment looks like this:

```
if @SearchCode = 'To workgroup'
BEGIN
  INSERT INTO OVBPI_ITEM_CHANGE_HIST
  (Object_id, EventTime, EventStatus, ItemType)
  VALUES (@Object_id, @CurrentTime, 'New', 'SC_Toworkgroup')
END
```

where the string `SC_Toworkgroup` is written in the `ItemType` column. The `ItemType` forms the last part of the OVBPI event name when the adaptor sends this event. By default, all the adaptors prefix the `ItemType` value with the string `ServiceDesk`. So the code segment above if effectively saying that the event to be generated is going to be `ServiceDesk/SC_Toworkgroup`.

You extend the other four triggers with similar SQL code segments each specifying different `ItemType` values.

In this worked example, the values chosen are:

`SC_Toworkgroup` for the Service Calls flow
`I_Toworkgroup` for the Incidents flow
`C_Toworkgroup` for the Changes flow
`P_Toworkgroup` for the Problems flow
`WO_Toworkgroup` for the Work Orders flow

You should apply these SQL code extensions to the template trigger files. The idea is to make your updates to the template files and use the `sd_createadaptors` script to generate the final set of files when everything is ready.

Extend the Template Adaptor Files

With the trigger writing additional records into the `OVBPI_ITEM_CHANGE_HIST` table, you need to extend the adaptor to handle these records.

You update the template adaptor configuration file and use `sd_createadaptors` to generate the final adaptor files when everything is ready. You choose the template file appropriate for the OVSD database your adaptor is interrogating.

All adaptor configuration files have a similar format. They consists of sections for each type of change they are monitoring. For example, in the Service Call adaptor there are sections for the components:

```
ServiceCallAdaptor.Component3.Name = StatusChange
ServiceCallAdaptor.Component4.Name = DeadlineChange
ServiceCallAdaptor.Component5.Name = PersonChange
ServiceCallAdaptor.Component6.Name = PriorityChange
```

Within each of these sections are all the configuration details for that component. So, to add a new component to monitor a new record type (`ItemType`) you essentially need to copy one of these component blocks and then modify this to handle the new record type.

The steps are basically the same for each adaptor. Let's consider how this is done for the Service Call adaptor:

1. Go to the end of the template adaptor configuration file.
2. Make a copy of the previous component's configuration section.
3. Increment the component number by 1.
4. Give this new component a unique name, and change this name throughout the whole component.
5. Change the primary key Select (`NextPrimaryKeySQL`) to look for the new `ItemType` value `SC_Toworkgroup`.
6. Work out which OVSD view you can use to get the Workgroup information and then use this in the Select statement (`SelectSQL`).

   You need to create tokens for any OVSD names and then define these within the `SD_Adaptor_Config.properties` file.
7. Save this adaptor configuration template file.

The configuration segment for the Service Call adaptor looks as follows:

```
ServiceCallAdaptor.Component7.Name            = WorkgroupChange
ServiceCallAdaptor.WorkgroupChange.LinkTo1    = ServiceCallAdornment

ServiceCallAdaptor.WorkgroupChange.ClassName  = org...jdbc.PollingSQLSource
ServiceCallAdaptor.WorkgroupChange.JdbcDriver = {{DB_DRIVER}}
ServiceCallAdaptor.WorkgroupChange.JdbcUrl    = {{DB_JDBC_URL}}
ServiceCallAdaptor.WorkgroupChange.Database   = {{DB_NAME}}
ServiceCallAdaptor.WorkgroupChange.UserName   = {{DB_USER_NAME}}
ServiceCallAdaptor.WorkgroupChange.Password   = {{DB_PASSWORD}}
ServiceCallAdaptor.WorkgroupChange.PasswordEncoding  = true
ServiceCallAdaptor.WorkgroupChange.ExitOnError       = false
ServiceCallAdaptor.WorkgroupChange.PollPeriod        = 60000
ServiceCallAdaptor.WorkgroupChange.QuoteMultipleKeys = true
ServiceCallAdaptor.WorkgroupChange.BatchSize         = 100
ServiceCallAdaptor.WorkgroupChange.UsingInClauses    = true
ServiceCallAdaptor.WorkgroupChange.PrimaryKeyRegExp  = PK

#
# get list of next primary keys to process
#
# Select the events of the new type "SC_Toworkgroup"
# ----------------------------------==============
ServiceCallAdaptor.WorkgroupChange.NextPrimaryKeySQL = SELECT buf.Object_id \
FROM OVBPI_ITEM_CHANGE_HIST buf \
WHERE \
buf.EventStatus = 'New' AND \
buf.ItemType = 'SC_Toworkgroup' \
ORDER BY buf.EventTime

#
# Process primary keys
#
ServiceCallAdaptor.WorkgroupChange.SelectSQL = SELECT \
buf.EventTime GeneratedDate, \
buf.ItemType EventName, \
sch.{{SCH_SERVICE_CALL_ID}} service_call_id, \
sch.{{SCH_CREATED_BY_DISPLAY_NAME}} changedBy, \
sch.{{SCH_DATE_CREATED}} changeTime, \
sc.{{SC_DEADLINE}} deadline, \
wkg.{{WORKGROUP_NAME}} workgroup \
FROM \
OVBPI_ITEM_CHANGE_HIST buf, \
{{HISTORY_LINE_SERVICE_CALL_VIEW}} sch, \
{{WORKGROUP_VIEW}} wkg, \
{{SERVICE_CALL_VIEW}} sc \
WHERE sch.{{SCH_OBJECT_ID}} IN ('PK') \
AND wkg.{{WORKGROUP_OBJECT_ID}} = sch.{{SCH_VALUE_TO}} \
AND sch.{{SCH_OBJECT_ID}} = buf.Object_id \
```

```
AND sc.{{SC_OBJECT_ID}} = sch.{{SCH_SERVICE_CALL_OBJECT_ID}} \
ORDER BY buf.EventTime

ServiceCallAdaptor.WorkgroupChange.CommitSQL  = {{COMMIT_SQL}}
ServiceCallAdaptor.WorkgroupChange.RollbackSQL  = update
OVBPI_ITEM_CHANGE_HIST set EventStatus='Error' where Object_id IN ('PK')
```

where:

- This section was a copy of component6.
- The component was then renamed to component**7**.
- The name was changed to **WorkgroupChange**.
- The `NextPrimaryKeySQL` is picking up the new `SC_Toworkgroup` entries.
- The `SelectSQL` has been altered to pull out the workgroup details.

The difficult part of putting together the adaptor configuration is the `SelectSQL`. This requires you to investigate the OVSD database and determine where the workgroup information is located. In this case, it is located in the single view `v_workgroup`. You tokenize this view name as it may vary depending on the way the OVSD database reporting views were set up.

Set Up New Tokens in the SD_Adaptor_Config File

Having worked out the name of the view your adaptor needs to access, and the column names it needs, you tokenize these names within the adaptor configuration template file, and set up these tokens in the `SD_Adaptor_Config.propertes` file.

```
WORKGROUP_VIEW        = v_workgroup
WORKGROUP_OBJECT_ID   = object_id
WORKGROUP_NAME        = name
```

Recreate the Adaptor Configuration Files and Trigger Files

You run `sd_createadaptors` to process your template files and reproduce a new trigger file and adaptor configuration files.

It is always worth while checking the resultant adaptor configuration files to ensure that there are no {{ }} items remaining. If there are then this means that you have either not defined them in your `SD_Adaptor_Config.properties` file, or mis-spelt them within your adaptor configuration files.

Remodel the flows

You need to remodel your flows to accept the new events.

1. Define the new data property in the associated data definition.

   The adaptors are sending in new data item(s). In this example, they are sending in a new data item called `workgroup`.

2. Define the new events

   The adaptors are sending in the following new events:

   `SC_Toworkgroup` for the Service Calls flow
   `I_Toworkgroup` for the Incidents flow
   `C_Toworkgroup` for the Changes flow
   `P_Toworkgroup` for the Problems flow
   `WO_Toworkgroup` for the Work Orders flow

   Each event contains the data:

   ```
   changedBy       (String[80])
   changeTime      (Date)
   deadline        (Date)
   workgroup       (String[80])
   ```

   and an appropriate ID field.

3. Define the new event subscriptions

   You must set up new subscriptions for each of the associated data definitions. They need to subscribe to the appropriate event, pulling out all the data fields.

4. Redeploy the flows

   The flows need to be redeployed.

Install the Modified Database Triggers

The extended SQL triggers need to be installed.

For Oracle, this is simply a case of re-installing the triggers (using the `CreateServiceDeskTriggers.sql` script) - this is because Oracle has the concept of "create or replace".

For an MSSQL database you need to first drop the triggers (using the `DropServiceDeskTriggers.sql` script) before then re-installing them (using the `CreateServiceDeskTriggers.sql` script).

Install and Run the New Adaptors

With the triggers in place, and the flows deployed, you are all set.

Start up the new adaptors and then try creating a new item within the SD Client. Assign this new item to a workgroup. You should see the adaptor sending the information across to OVBPI.

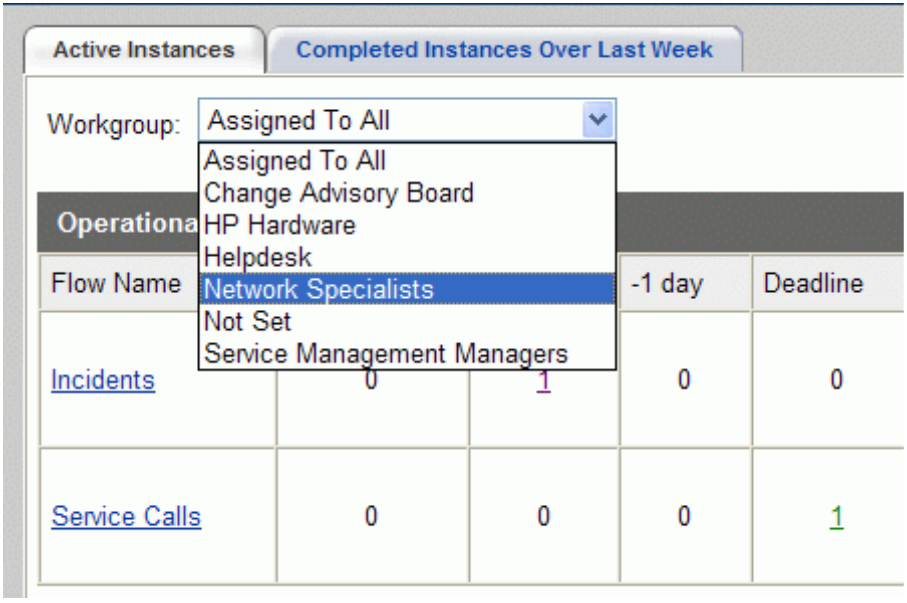
You can use the existing custom dashboard to see if the new workgroup information is getting across to OVBPI. By drilling into a particular instance you should be able to see all the associated data - including the workgroup.

Remember that the OVBPI flows only track an item once it has received a state change. That is, if you modify an existing item and alter the workgroup, you do **not** see this item reflected in OVBPI. You must first change the state of the item (that is, change it from `Registered` to `In progress`). Then all workgroup assignment changes for this item is tracked.

Customize the Dashboard to allow users to filter by workgroup.

Once you have workgroup information coming across to OVBPI, you can customize the dashboard further such that users can filter the items on the main screen not by operator....but by workgroup (or both)!

For example, you could customize the main screen as follows:



where:

- Instead of selecting by assigned operator, the user can select by assigned workgroup.
- When the user has selected a workgroup, the main page then only shows items currently assigned to that workgroup.

To customize the dashboard requires that you are comfortable reading and modifying JSP and Java code.

6

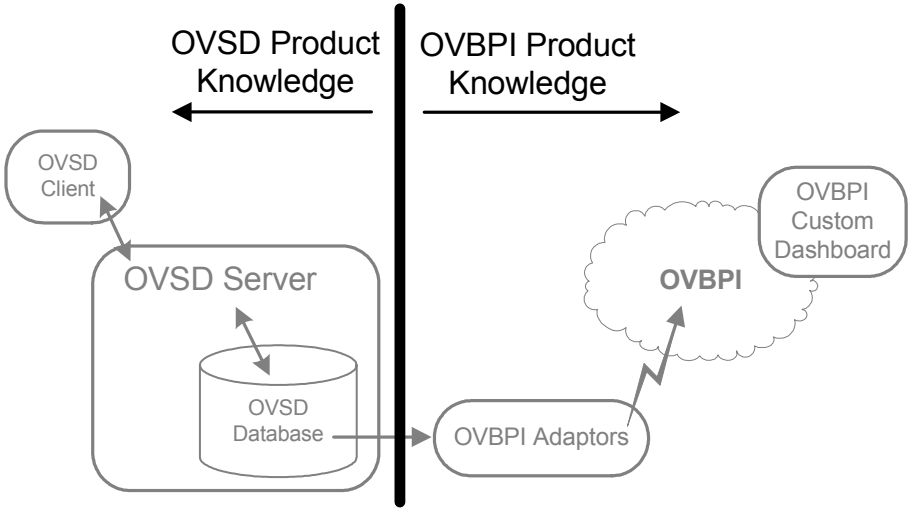
Further Topics

This chapter includes sections on troubleshooting the OVSD Process Insight module.

Troubleshooting

If you think about the division of product knowledge required to support the OVSD Process Insight module it can be essentially divided as follows:

Figure 15 Product Knowledge Division



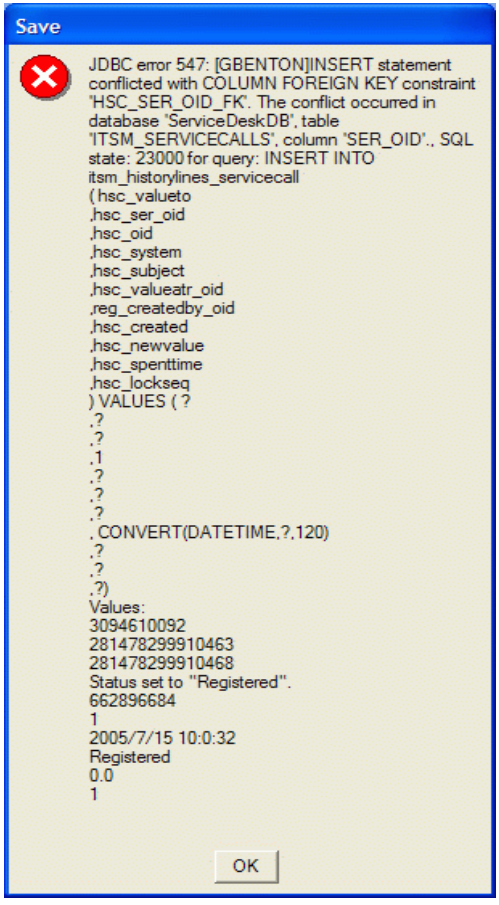
Certainly if someone has a problem using the OVBPI custom dashboard then it is an OVBPI issue. However the situation may not be that straightforward if someone using the SD Client complains that they cannot add a new Service Call.

Let's now consider some example problems and their solutions.

JDBC Error in the SD Client

Because the OVSD Process Insight module adds triggers to the history line tables, if there is a problem with a trigger on one of the history lines tables this may prevent the entering or updating of the corresponding item (Service Call, Incident, Change, Problem or Work Order).

For example, if there is a problem with the logic in the trigger placed on the `ITSM_HISTORYLINES_SERVICECALL` table, when the SD Client tries to add a new Service Call, or update an existing Service Call, they receive a JDBC Error. The message may look something like this:



Unfortunately, there is nothing obvious in this message to help you decide whether the problem is due to the OVBPI SQL trigger. The message is basically saying that “Something went wrong when doing the SQL Insert”.

Do not simply assume that this error message means that there is a problem with the SQL trigger. There may be a valid OVSD problem totally unrelated to the trigger. But certainly the trigger can cause the SD user to be unable to add or update items (Service Calls, Incidents, etc.).

Install Testing

Clearly you can avoid any such JDBC problems by ensuring that the person who installs the SQL Triggers tests that an SD Client is able to add and update Service Calls, Incidents, etc.. once the triggers in place.

Access to the OVBPI Table

If your triggers are causing your JDBC problems, make sure you also check that the `OVBPI_ITEM_CHANGE_HIST` table is correctly created and that your triggers are able to access this table. The default installation sets this up correctly, however it is always worth checking if your triggers are having a problem.

Dropping the Triggers

This is **not** something that you really want to do on a production system as it causes OVBPI to miss activity within OVSD. However, if you are getting these JDBC problems when adding or updating items within OVSD, you could choose to drop all the OVSD Process Insight module triggers. This must only be considered when all other options have been exhausted, as it causes an interruption to the OVBPI monitoring.

See Removing Process Insight Triggers on page 37.

Obviously, once the problem has been resolved you would want to reinstate the corrected triggers so as to allow OVBPI monitoring to continue.

Run Time Adaptor Errors

A very important step in setting up the adaptors and triggers is the step where you set up the `SD_Adaptor_Config.properties` file. If you do not get every name in this file correct for your installation then your adaptors have trouble...but they are most likely to have this trouble when they come to process their first data record. That is, the adaptors may well start up successfully, but it is not until they have each processed a record that you can be sure you have everything configured correctly.

Setting the `SD_Adaptor_Config.properties` file is one of the most important steps in the whole process. So take your time when doing it. Make sure every name corresponds to the correct name in your OVSD database.

If you do have a mis-naming problem then your adaptors throw an exception **at run time**. Here is an example exception thrown by the Service Call adaptor:

```
[05/10/22 14:22:46.719] INFO: StatusChange executing query [SELECT
 buf.EventTime GeneratedDate, buf.ItemType EventName, sch.service_call_id
 service_call_id, sch.service_call_description description,
 sch.created_by_display_name changedBy, sch.created changeTime, rct.rct_name
 valueTo, sc.service_name serviceName, sc.service_id serviceId FROM
 OVBPI_ITEM_CHANGE_HIST buf, v_history_line_incorrectViewName sch,
 rep_codes_text rct, v_service_call sc WHERE sch.object_id IN
 ('281478296240215') AND rct.rct_rcd_oid = sch.value_to AND
 sch.object_id = buf.Object_id AND
 sc.object_id = sch.service_call_object_id ORDER BY buf.EventTime]
[05/10/22 14:22:46.719] WARN: SQL threw exception, com.inet.tds.SQLException:
 Msg 208, Level 16, State 1, Line 1, Sqlstate S0002[GBENTON]Invalid object
 name 'v_history_line_incorrectViewName'.
[05/10/22 14:22:46.719] FATAL: Unexpected exception, Source terminating,
 java.lang.NullPointerException
[05/10/22 14:22:46.719] INFO: ServiceCallAdaptor.Controller - Source
 StatusChange is exiting
[05/10/22 14:22:46.729] INFO: Adaptor terminating with error code 1
```

The main cause of the run-time error is that the name of one of the OVSD Database Reporting Views is incorrect for this installation.

To fix this problem:

- Fix the naming error in the `SD_Adaptor_Config.properties` file
- Re run `sd_createadaptors` and recreate all the adaptor configuration files.
- Re run the adaptor(s).

Where Are My Flow Instances?

If you have made changes to an item in OVSD (using the SD Client) but nothing is appearing in the OVBPI dashboard then it could be because of a number of different reasons:

- Check that the adaptors are up and running
- Check that OVBPI is up and running and has not reported any errors
- Check that the SQL triggers are installed in the OVSD database

It may be that you know everything is up and working because when you make a change through the SD Client you can see the adaptor sending information to OVBPI...yet no flow instance gets created!

If everything looks alright and you are still unable to understand why a certain item does not appear in the OVBPI dashboard, it might be because the item has **not yet changed its state**. Until an item has changed its state, that item does not appear in OVBPI. See What The Dashboard Displays? on page 41 for more details.