

HP

Business Process Insight

For the Windows® Operating System

Software Version: 7.50

Integration Training Guide - Defining Business Process Monitors

Document Release Date: June 2008

Software Release Date: June 2008



Legal Notices

Warranty

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

The information contained herein is subject to change without notice.

Restricted Rights Legend

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

Copyright Notices

© Copyright 2006 - 2008 Hewlett-Packard Development Company, L.P.

Trademark Notices

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

Microsoft® is a US registered trademark of Microsoft Corporation.

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

UNIX® is a registered trademark of The Open Group.

Documentation Updates

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

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

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

<http://h20230.www2.hp.com/selfsolve/manuals>

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

Support

You can visit the HP Software Support web site at:

www.hp.com/go/hpsoftwaresupport

HP Software online support provides an efficient way to access interactive technical support tools. As a valued support customer, you can benefit by using the support site to:

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

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

To find more information about access levels, go to:

www.hp.com/managementsoftware/access_level

To register for an HP Passport ID, go to:

www.managementsoftware.hp.com/passport-registration.html

Contents

1 Business Monitors	9
Introduction	10
Monitor Definer	11
Running the Monitor Definer	11
Logon User/Password	12
Deployed Processes	12
Creating a Business Monitor	13
The Process Diagram	15
Monitor Name	15
Monitor Description	15
Monitor Scope	15
Monitor Type	16
Statistics Collection	17
Filters, Groups and Deadlines	23
Apply Filter	23
Group Results By	24
Deadline Property	25
Filters	26
Creating a Filter	26
Example Filters	27
Modifying a Monitor	29
Creating a Threshold	30
Threshold Type	31
Warning/Minor/Major/Critical Violation	36
Violation Message	36
Violations	36
Instance and Statistical Thresholds/Violations	36
Violation Levels	37

Options	38
Refreshing the Monitor Definer	38
Exporting Monitors	38
Importing Monitors	39
Monitor Definer Help	40
Lab - Defining Monitors	41
The Scenario	41
Defining the Process	42
Understanding the Process	43
Defining the Monitors	44
The Process Simulator	51
Running the Call Center	51
End of The Lab	56
2 Monitor Engine	57
How Monitors Work	58
Monitor Events	58
Monitor Values	59
Monitor Statistics	61
Monitor Violations	63
Monitor Notifications	64
The Big Picture	66
Violation Timings	67
Instance Thresholds	67
Statistical Thresholds	68
Monitor Engine Off/Restart	69
Violations	69
Statistical Monitors ("Back Filling")	69
Instance Cleaner Settings	71
Monitor Instance Cleaner	71
Business Impact Engine Instance Cleaner	71
3 Custom Monitors	73
Monitor Scope	74
Defining a Custom Monitor	75
The Stored Procedure	75

The Monitor Definer	80
Example - A Data Property	82
The Stored Procedure	83
The Custom Monitor Type Definition	87
Example - Percentage Path Process	88
Monitor Scope	89
The Stored Procedure(s)	89
The Custom Monitor Type Definitions	95
Defining The Monitors	97
Defining Thresholds	98
The HPBPI Business Process Dashboard	99
SQL Errors	100
HPBPI on Oracle	100
HPBPI on MSSQL	100
Lab - Custom Monitors	101
The Call System Process	101
The Required Monitors	101
4 Further Topics	105
Monitor/Threshold Activation	106
Pre-Dated Events (BPI_GeneratedDate)	106
Detecting Thresholds	107
Instance Violations	109
Deadline Monitor Value is Fixed	109
Redeploying Processes/Monitors	110
Superseded Processes/Monitors	111
Deleting Process Monitors/Thresholds	111
Avoiding Notification Storms	112
No Collection Interval Defined	113
Business Process Dashboard and Violations	114
Business Health Scorecard Page	114
Upgrading Custom Monitors	115
The Original Interface	116
Version 2 of the Interface	117
Upgrading a Custom Monitor to use Version 2	118

1 Business Monitors

This chapter looks at how to create and use business monitors using the HP Business Process Insight (HPBPI) Monitor Definer.

Introduction

Once an HPBPI process is deployed and running, the HPBPI Business Impact Engine maintains basic statistics about the process, such as:

- The overall state of the process.
- The number of active process instances.
- Specific process instance statistics such as:
 - The start and stop times of each step.
 - Step durations.

These statistics are held in the HPBPI database.

In addition to the standard process and step statistics that are maintained for all processes, by the Business Impact Engine, you can specify additional process statistics, known as **business process monitors**. For example, you can measure business process monitors such as:

- The time taken to process an order.
- The current backlog of flights waiting to get airborne.
- The average value of an order.

Once you have configured one or more business process monitors for a process, you can configure **thresholds**. This allows HPBPI to raise a violation when a business process monitor exceeds one of your thresholds. For example, you may configure the following thresholds:

- Issue a violation when the time taken to process an order exceeds four hours.
- Issue a warning violation when the current backlog of flights waiting to get airborne is greater than 10. Issue a critical violation when this number exceeds 50.
- Issue a violation when the value of a particular order is more than two standard deviations greater than the normal average order. In other words, issue a violation when a particularly oversized order is in the system.

Once defined, your monitors, thresholds and violations are reported in the HPBPI Business Process Dashboard.

Monitor Definer

Once a process is defined and deployed, you use the HPBPI Monitor Definer to define monitors and thresholds for that process.

Running the Monitor Definer

The Monitor Definer runs within a Web browser, thus you need to make sure that you have the HPBPI Servlet Engine running, and that the HPBPI database is up and running.

To run the Monitor Definer you can either go to:

Start->Programs->HP->HP Business Process Insight->Monitor Definer

or, start a Web browser with the URL:

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

where:

- *hostname* is the hostname of your HPBPI installation
- *44080* is the default port for the Servlet Engine

For example:

```
http://localhost:44080/ovbpimetricdefiner
```



The HPBPI Monitor Definer used to be called the Metric Definer. That is why the URL for running the Monitor Definer contains the text “metricdefiner”.

Logon User/Password

When you run the Monitor Definer, it asks you to log on as a valid user.

The default user name/password details are:

```
User:      admin
Password: hpbpi
```

To change the admin user password, refer to the *HPBPI System Administration Guide*.

Deployed Processes

You can create monitors only for processes that are already deployed.

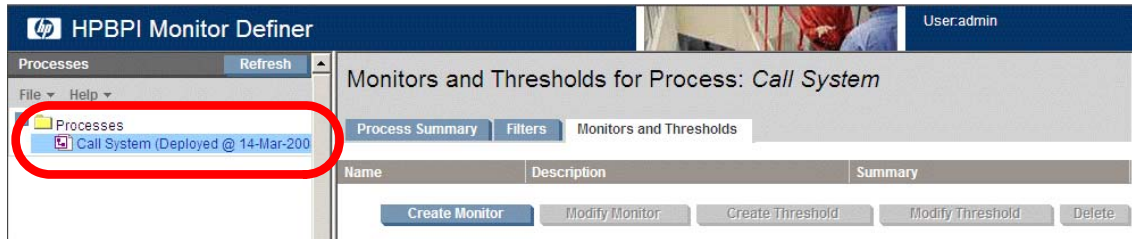
If while you are running the Monitor Definer, a new process is deployed, you can click on the `Refresh` button from within the left-hand Navigator frame of the Monitor Definer (not the Web browser's toolbar). This refreshes the list of deployed processes available to you within the Monitor Definer.

Creating a Business Monitor

To create a business monitor you must first select the process. You simply click on the required process in the left-hand Navigator frame within the Monitor Definer.

For example, consider [Figure 1](#):

Figure 1 Monitor Definer - Selecting a Process



When the process is selected, the right-hand frame gives you three tab options:

- Process Summary

Clicking this tab gives you a picture of the process definition and some basic details of the process - such as description and deployment date.

- Filters

This tab allows you to create and modify filters. Filters are explained in more detail in the section [Filters](#) on page 26.

- Monitors and Thresholds

This tab allows you to create/modify monitors and create/modify thresholds. This is the default tab when you click on a process in the left-hand frame.

Once a process is selected, you click on the `Create Monitor` button, in the right-hand frame. You are then presented with the details you need to complete to create a new monitor.

For example, consider [Figure 2](#) on page 14.

Figure 2 Monitor Definer - Create a new Monitor

Process Name: Call System (Deployed @ 14-Mar-2008 15:18:38)

Monitor Name: A unique name for your Monitor

Monitor Description:

Monitor Scope: Select a scope to reveal relevant options

Monitor Start Step:

Monitor Type:

Statistics Collection: On Off
Collection Interval: Minutes Hours Days

Collected Data:

Apply Filter:

Group Results By:

Deadline Property:

Let's now consider each of the fields in more detail...

The Process Diagram

The process diagram is displayed and two monitor flags (red colored flags) are drawn on one of the steps. These monitor flags define the start and end of the monitor you are defining. Initially these flags are placed on a single step, and as you define the monitor more fully, these flags move to reflect your configuration.

Monitor Name

You need to give your monitor a unique name. This name must be unique across all your monitors.

Monitor Description

You can provide your monitor with some description text. This is optional.

Monitor Scope

The following options are available to you.

Whole Process

Selecting this option causes the monitor flags to disappear.

Setting a `Whole Process` monitor means that when a process instance first starts (at whatever step actually starts the process instance), a monitor instance is started. This monitor instance completes when the process instance completes.

Single Step

Setting a `Single Step` monitor means that a monitor instance starts when a process instance enters the specified step. This monitor instance completes when the process instance exits this same step.

Monitor Start Step

When selecting a `Single Step` monitor, you are also asked to specify the step itself. As you specify this step, the monitor flags within the process diagram move to highlight this step.

Multiple Steps

Setting a `Multiple Steps` monitor means that this monitor measures between two steps within the process diagram.

Monitor Start Step

You select the start step for your monitor. You specify whether it is the start or completion of this step that starts each monitor instance.

Monitor End Step

You select the end step for your monitor. You specify whether it is the start or completion of this step that completes each monitor instance.

Monitor Type

Here you select the type of monitor. There are two monitor types available by default:

- **Duration**

The monitor measures a duration. You measure the time taken from the start of the monitor to the end of the monitor, as specified in the scope of the monitor.

- Process Value

The monitor measures the process value. The process value is the value contained in the process value attribute as defined within the process definition. For example, you may wish to measure the value of each order (assuming that your process definition holds the order value within the process value attribute).

You have the capability of adding your own Monitor Types, known as Custom types. (See [Chapter 3, Custom Monitors](#) for more details and examples.)

Statistics Collection

When creating a monitor, the “Statistics Collection” option within the dialog alters depending on whether or not you have configured the Business Availability Center data sample integration with HPBPI.

If you have not configured the Business Availability Center data sample integration with HPBPI, then the statistics collection option looks as shown in [Figure 3](#) (and in [Figure 2](#) on page 14), where you are able to turn statistics collection on or off and then provide a value for the collection interval.

Figure 3 Statistics Collection

Monitor Start Step: Assigned

Monitor Type: Duration

Statistics Collection: On Off

Collection Interval: 15 Minutes Hours Days

Collected Data: Backlog - count

- Backlog - process value
- Throughput - count per hour
- Throughput - process value per hour
- Average duration
- Minimum duration

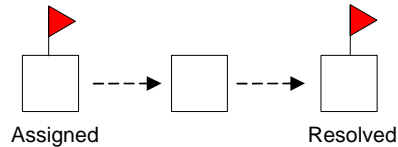
Let's first consider how statistics collection works, and then discuss the differences when the Business Availability Center data sample integration with HPBPI is configured - see [Business Availability Center Integration and Collection Intervals](#) on page 21.

Recording Monitor Values

Selecting the `scope` of the monitor determines when the actual monitor instance value is recorded.

For example, suppose you configure a monitor, and set the monitor scope to be from the start of the step `Assigned` to the end of the step `Resolved`, as shown in [Figure 4](#).

Figure 4 Example Monitor Definition



When a process instance enters the `Assigned` step, a new monitor instance is instantiated and HPBPI records the time this monitor instance starts. If the monitor is defined as a `process value monitor`, then HPBPI also records the current process value.

When this process instance eventually completes the `Resolved` step, HPBPI completes the associated monitor instance and records the time it completes. If the monitor is defined as a `duration monitor`, then HPBPI calculates the overall duration that this monitor instance has taken to complete. If the monitor is defined as a `process value monitor`, then HPBPI records the final process value of this monitor instance.

As each monitor instance starts, HPBPI records the start time for the monitor instance. As each monitor instance completes, HPBPI determines the duration, or process value, for the monitor. This duration, or value, represents the “value” for this completed monitor instance. So the “value” of the monitor instance is recorded at the completion of each monitor instance, as defined by the scope of the monitor.

Calculating Statistics

If the “value” of the monitor instance is recorded at the completion of each monitor instance, as defined by the scope of the monitor, then why do you need to enable `Statistics Collection`?

Yes, each monitor instance value is recorded as the monitor instance completes. But what about being able to calculate the average of all your orders? or whether a particular incoming order rate is faster or slower than normal?

To be able to calculate things such as averages, throughput rates and backlogs, you need to enable Statistics Collection.

Collection Interval

If you enable Statistics Collection, you need to specify how often you want the statistics to be calculated. This is specified by the `Collection Interval`.

If you were to specify a collection interval of 15 minutes, then every 15 minutes HPBPI would look back through the last 15 minutes of monitor data values that had been collected and produce statistics. These statistics would include calculations such as the average of all monitor instances completed in the last 15 minutes, the minimum/maximum/standard deviation and throughput of all the monitor instances completed within the last 15 minutes, and the count of all monitor instances still active at the end of the last 15 minute period (also known as the `backlog`). As you will see in a later section ([Creating a Threshold](#) on page 30) you can then set thresholds against these statistics such that (for example) if the average time to complete an order for the last 15 minutes is more than two standard deviations slower than the normal average for completing orders, then raise a violation.

Collection Interval Timings

Suppose the time is 11:12am and you have just created a new monitor and specified a collection interval of 15 minutes. As you create the new monitor, within the Monitor Definer, it is activated within HPBPI the moment you press the `OK` button. So the question is, does the collection interval run every 15 minutes from 11:12 (the time you created the monitor)? Answer.....No!

When you create a monitor and specify a collection interval, HPBPI wants to align the collection interval to an “easily understandable” time slot. That is, if you specify a collection period of 15 minutes, it makes sense to run the collection interval on the hour and at 15, 30 and 45 minutes past the hour. To achieve this, HPBPI typically shortens the first collection interval so that they can be aligned within the hour.

So, in this example, where you defined the monitor at 11:12am, the collection intervals runs as follows:

- 11:12 -> 11:15
- 11:15 -> 11:30
- 11:30 -> 11:45
- 11:45 -> 12:00
- etc...

where the first collection interval is less than the specified 15 minutes.

If you specify a collection interval of one day then the first collection period ends that night at midnight. The intervals then run from midnight to midnight.

If you specify a collection interval of one week then the first collection interval ends the following Saturday night at midnight. The week then runs from Saturday midnight to Saturday midnight.

Choosing a Collection Interval

The time you specify for the collection interval needs some thought.

For example, suppose you want to track the average throughput of your orders on a daily basis. That is, you want to calculate your average daily order rate. You could set the collection interval to be one day and this would give you exactly what you want. However, be aware that this order throughput is only calculated each collection interval. That is, the throughput is calculated at the end of each day. Thus you would not see the order throughput rate for today until midnight tonight. And this throughput rate is not changed until the following day (at midnight). That is, each day it shows you yesterday's order throughput rate.

Now, suppose that you want to track the average throughput rate of your orders such that you can set a threshold that raises a violation whenever you start to see orders going through your system too slowly. Setting a collection interval of one day is probably not appropriate. With a collection interval of one day you do not see any statistical data until the next day. Also, if there is a slow down of orders that lasts for maybe one hour, this may not have any real effect on the overall throughput figures for that day. You probably want to set a smaller collection interval - maybe one hour, maybe less.

So the collection interval determines how frequently HPBPI calculates monitor statistics, and these statistics are a view of what has happened over the defined collection interval.

Do You Need To Collect Statistics?

You may not need to collect any monitor statistics.

For example, you may define a monitor that simply measures the time taken for each of your orders to be processed. You may then have a threshold that raises a violation whenever any individual order takes more than a certain time to complete. In this situation you do not require any averages, throughputs or standard deviations...you are simply measuring and testing the time taken on a individual order basis.

If your monitor is simply measuring the duration (or process value) of process instances then you do not need to collect statistics. HPBPI is able to measure thresholds and raise violations based purely on the active and completed monitor instances.

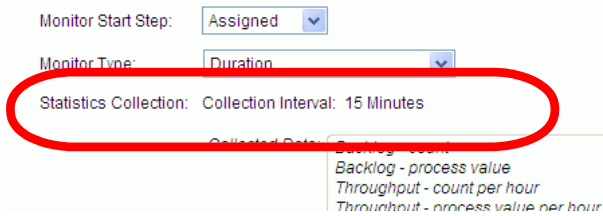
An example might be where you wish to have a monitor that measures the time it takes for you to close your priority-1 support calls. If any call is taking longer than four hours then raise a violation. For this sort of monitor and violation scenario you do not need to collect statistics. If on the other hand you wanted to measure whether the average call resolution time for priority-1 call is faster or slower than the normal average resolution time, then you need to collect statistics.

Business Availability Center Integration and Collection Intervals

When you have HPBPI configured to send data samples to Business Availability Center, the statistics collection setting within the Monitor Definer is already configured for you. You are not able to set the statistics collection interval from within the Monitor Definer.

For example, when you are creating a monitor, the statistics collection part of the dialog might look as shown in [Figure 5](#) on page 22.

Figure 5 Statistics Collection with BAC Data Samples Configured

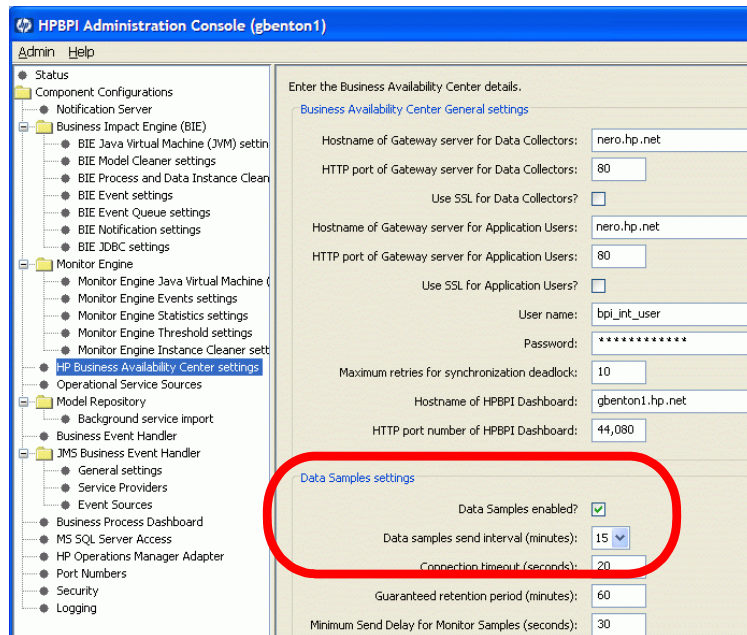


Notice that the collection interval is fixed for this monitor.

The collection interval shown for the monitor corresponds to the Data sample send interval that is configured for the Business Availability Center settings within the HPBPI Administrator Console - refer to the *HPBPI System Administration Guide* for details of how to configure the Business Availability Center settings.

For example, the collection interval shown in Figure 5 is set to 15 minutes. This is because the Data samples send interval that is configured within the HPBPI Administration Console for the Business Availability Center data samples settings, has been set to 15 minutes, as shown in Figure 6.

Figure 6 HPBPI Admin Console - BAC Data Samples Send Interval



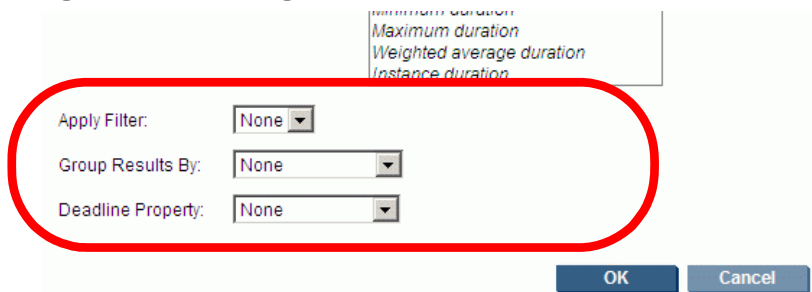
So when you have the integration to Business Availability Center configured within the HPBPI Administration Console, the collection interval for all your monitors is set to the time period specified for the `Data samples send interval`.

If you change the data samples send interval within the HPBPI Administration Console, then the collection interval of all your monitors is automatically updated to this newly configured send interval time period. This update of the monitor collection interval occurs the moment you apply the changes within the HPBPI Administration Console - there is no need to restart the Monitor Definer or any HPBPI services.

Filters, Groups and Deadlines

Lets consider the remaining options you can specify when creating a monitor as shown in [Figure 3](#) on page 17. The remaining options are shown again here in [Figure 7](#).

Figure 7 Creating a new Monitor - continued



The screenshot shows a dialog box for creating a new monitor. At the top, there is a list of duration types: `Maximum duration`, `Weighted average duration`, and `Instance duration`. Below this, three dropdown menus are visible, each with the text 'None' and a downward arrow. These are labeled 'Apply Filter:', 'Group Results By:', and 'Deadline Property:'. A red oval is drawn around these three dropdown menus. At the bottom right of the dialog, there are two buttons: 'OK' and 'Cancel'.

Apply Filter

The `Apply Filter` option lets you select a filter to apply to your monitor definition. You must have first created a filter before trying to select it when creating (or modifying) a monitor.

By selecting a filter, you are able to restrict the number of monitor instances recorded. For example, you might apply a filter that filters out only those orders placed by gold customers.

When applying a filter to a monitor you must make sure that the attribute(s) tested within your filter actually have values at the time each monitor instance is started. Otherwise, your filter always evaluates to false and no monitor instances ever start.

See [Filters](#) on page 26 for more details about creating monitor filters.

Group Results By

The `Group Results By` field allows you to select one attribute from the process's related data definition. The monitor data and statistics are then collected and grouped by the values within that data attribute.

For example, suppose you are defining the monitors for an airport that has four flight terminals. You might choose to group the results by the airport terminal. This allows you to see at a glance the performance for each terminal.

The attribute you select for `Group Results By` must be a `String` attribute.

Group Name

If you select a `Group Results By` attribute, the Monitor Definer displays an additional field called `Group Name`. This allows you to specify a display name to be used within the HPBPI Business Process Dashboard when displaying the monitor data in groups.

Within the HPBPI Business Process Dashboard you are able to display the monitor data in overall terms, or in groups. When displaying the monitor data in groups the HPBPI Business Process Dashboard uses your `Group Name` text in the heading of the graphs.

Deadline Property

The `Deadline Property` field allows you to select an attribute of data type date, from the process's related data definition.

By selecting a `Deadline Property` you give the monitor the ability to provide a `Deadline threshold`. You are able to specify a threshold that raises a violation if a monitor instance is still running relative to the date value specified within your `Deadline Property` attribute.

For example, suppose you have a process that measures mortgage applications. You want to set a threshold that raises a violation if a mortgage does not reach a certain step in the process within 20 days of the mortgage application date. By setting the `Deadline Property` to a data attribute, which contains the mortgage application date within the data definition, you can specify such a threshold.

Filters

When creating, or modifying, a monitor you can apply a filter. But before you can select a filter you must have defined one.

Creating a Filter

A filter consists of an expression, and the form of the expression is the same as that used when defining event subscription filters within the HPBPI Modeler (see *HPBPI Reference Guide* for the details of the filter expression syntax).

For example, a filter of:

```
data.Customer_Type == "Gold"
```

filters only those instances where the `Customer_Type` data attribute is set to the string value `Gold`.

Within the Monitor Definer, once you have selected your process (in the left-hand navigator frame), you then click on the `Filters` tab in the right-hand frame. You then click the `Create Filter` button and you can create your filter.

An example filter creation screen might look as shown in [Figure 8](#).

Figure 8 Monitor Definer - Filter Creation

The screenshot shows a dialog box titled "Monitor Definer - Filter Creation". At the top, it displays "Process Name: Call System (Deployed @ 14-Mar-2008 15:18:38)". Below this, there are three main sections: "Filter Name:" with a text input field containing "New Filter" and a tooltip "A unique name for your Filter"; "Filter Description:" with an empty text area; and "Data Definition:" which features a tree view of data properties under "Calls/Data", including "Call_Priority : String [2]", "Customer_ID : String [30]", "Call_Entry_Date : Date", and "Engineer_Name : String [50]". A tooltip "Select a Data Definition property to paste" is visible next to this section. Below the tree view is a "Paste" button. At the bottom, there is a "Filter Expression:" text area and "OK" and "Cancel" buttons.

Let's consider each of the fields in more detail...

Filter Name

When you define a filter you assign it a name. This must be unique across all the filters defined for this process.

Filter Description

You can provide description text for this filter.

Filter Expression

You are required to provide a filter expression. This filter expression involves the testing of data within the process's related data definition and must produce a true or false result.

To help you formulate your filter expression, the process's related data definition attributes are displayed for you in the `Data Definition` text box. You can select an attribute within the `Data Definition` text box and click the `Paste` button. Your expression can involve multiple data attributes and you can use AND, OR logic.

All data attribute names are prefixed by the name of the data definition relationship as defined in the process's definition. That is, if the related data definition is related using the name `data` then the attributes are specified as **`data.attributeName`**.

When defining a monitor filter, make sure that data attribute(s) you use within your filter expression have values at the start of the monitor when each process instance is running. That is, you cannot filter on (for example) `Customer_Type == "Gold"` if at the start of the monitor, the process instance has not yet set the `Customer_Type` attribute to a value.

Example Filters

```
data.Priority == 1
```

Filters only those instances where the `Priority` attribute value is set to the number 1 (one).

```
data.Status == "1"
```

Filters only those instances where the `Status` attribute value is set to the string value `1` (one).

```
data.StateValue.in("1", "2", "3")
```

Filters only those instances where the `StateValue` attribute is a string value that is in the set of values `1`, `2`, or `3`.

```
data.FlightCarrier.starts("QF")  
|| data.FlightCarrier.starts("BA")
```

Filters only those instances where the `FlightCarrier` attribute value starts with either the string `QF` or `BA`.

```
data.FlightCarrier.starts("QF")  
&& data.ClientType.contains("Gold")
```

Filters only those instances where the `FlightCarrier` attribute value starts with the string `QF` and the `ClientType` attribute value contains the string `Gold`.

Modifying a Monitor

Once you have defined a monitor for a process, it is listed in the right-hand frame of the Monitor Definer, along with any other monitors you have defined.

To **view** the detailed definition of a monitor you select the monitor (by clicking on the monitor definition in the right-hand frame) and then click on the `Modify Monitor` button. This shows you the full monitor definition and allows you to view, or modify, the monitor. Once you have seen the definition of the monitor you can click the `Cancel` button and the monitor is unchanged. As well as the `Cancel` button there are two other options: `Update` and `Replace`.

Once you have defined a monitor within the Monitor Definer that monitor is active within HPBPI and collecting monitor data. If you decide that you wish to make a modification to the monitor definition you have two choices:

- **Update** the monitor

You can choose to make the modification to the monitor but keep all the previously collected monitor data. This is achieved by using the `Update` option.

By pressing the `Update` button you are telling HPBPI to keep all currently collected monitor data, and to start collecting new data according to the newly modified monitor definition.

You obviously need to be careful about using the `Update` button. If you are simply modifying the description of the monitor definition then the `Update` button is the button to use. However, you need to think carefully if you are modifying the monitor to add (for example) a filter, as this means that any previously collected data and the newly collected data might represent different things, and this may lead to confusion when you display all the monitor data together.

- **Replace** the monitor

If you are modifying the monitor in such a way that it does not make sense to keep the previously collected data, then you should use the `Replace` option. For example, if you are modifying an existing monitor definition such that it now has a filter, it may not make sense to keep any previously collected monitor data. By modifying the monitor definition and clicking `Replace`, you are telling HPBPI to delete all previously collected monitor data for this monitor definition, and threshold violations, and start collecting afresh.

Creating a Threshold

Once you create a monitor, the monitor data is collected and stored in the HPBPI database. The HPBPI Business Process Dashboard allows you to report on this data.

As well as defining monitors, the Monitor Definer also lets you define (create) **Thresholds**. For example, suppose you have a monitor that is measuring the time it takes for you to process your orders. You may wish to set up a threshold to alarm when any individual order is taking more than four days to be processed.

Within the Monitor Definer, once you have created a monitor, you are then able to create one or more thresholds against this monitor.

To create a threshold:

- Select the process definition - in the left-hand navigator frame
- Select the monitor definition - in the right-hand frame
- Click the Create Threshold button...

The screen appears in the right-hand frame, to create a new threshold. An example screen might appear as shown in [Figure 9](#).

Figure 9 Monitor Definer - Create Threshold

Process Name: Call System (Deployed @ 14-Mar-2008 15:18:38)

Monitor Name: Call Assignment Time

Monitor Type: Duration

Monitor Scope: Single step

Threshold Name:* *A unique name for your Threshold*

Threshold Description:

Threshold Type:* *Select a type to reveal relevant options*

The process and monitor details are automatically filled in for you.

You provide a name for this threshold. This name must be unique across all the thresholds for this process.

You can then (optionally) provide a description for this threshold.

You then select the `Threshold Type` from the pull-down list, and this determines the remainder of the screen details. Let's consider the remaining options for this screen...

Threshold Type

The list of available threshold types alters depending on whether you have the data samples integration to Business Availability Center configured within the HPBPI Administration Console.

Let's first consider the complete list of threshold types available within the Monitor Definer, and then discuss in more detail the differences when the Business Availability Center data sample integration with HPBPI is configured - see [Business Availability Center Integration and Threshold Types](#) on page 35.

The complete list of threshold types available within the Monitor Definer are as follows:

- Absolute Duration/Value
- Backlog

This threshold type is not available when the integration to Business Availability Center is enabled.

- Deadline
- Relative Duration/Value
- Throughput

This threshold type is not available when the integration to Business Availability Center is enabled.

Absolute (Duration/Process Value/Value)

An absolute threshold allows you to set a threshold against an absolute value.

The text for the threshold type, and the units available on the threshold definition page, are dependent on the `Monitor Type` as defined for the underlying monitor. If the `Monitor Type` for the monitor is `Duration`, then the threshold type is called `Absolute duration`. If the `Monitor Type` for the monitor is `Process Value`, then the threshold type is called `Absolute`

process value. If the `Monitor Type` for the monitor is a custom defined monitor (see [Chapter 3, Custom Monitors](#)) then the threshold type is called Absolute value.

Threshold Measure

You can set the threshold measure to be any specific instance, or a recent average/minimum/maximum.

- Any Specific Instance

This allows you to set a threshold to raise a violation when any monitor instance takes longer/shorter than the set time. Or when the process value of any monitor instance is greater/smaller than a set value.

This threshold measure means that the moment a monitor instance exceeds the threshold, a violation is triggered. For example, if your threshold is measuring that orders take less than four hours to complete, the moment an order has been running for four hours the threshold raises the violation.

- Recent

This allows you to set a threshold to alarm when the recent average (or minimum/maximum) is greater/smaller than a set value.

The term `recent` means the value calculated over the last collection interval of your underlying monitor. If the collection interval is 15 minutes then this threshold checks for any violations every 15 minutes.

This option is not available when the integration to Business Availability Center is enabled.

Backlog

The backlog threshold type measures the number of monitor instances that are active at the end of each collection interval for the underlying monitor. So, setting a backlog threshold type means that the value is tested at the end of each collection interval.

This threshold type is not available when the integration to Business Availability Center is enabled.

Threshold Measure

You can test the `Count` of active monitor instances, or the `Total process value` of active monitor instances.

Deadline

The `Deadline` threshold type is available only if the underlying monitor definition specified a `Deadline Property` attribute.

Setting a deadline threshold type allows you to measure that any monitor instance has reached a deadline.

For example, you might say that you wish to raise a violation if a monitor instance is still running and the current time is 20 days after the date/time specified in this instance's `Deadline Property`. Or you might like to raise a violation if a monitor instance is still running and the current time has reached the date/time specified in this instance's `Deadline Property`.

Relative (Duration/Process Value/Value)

A relative threshold allows you to set a threshold against the standard deviation of your collected monitor values.

The text for the threshold type is dependent on the `Monitor Type` as defined for the underlying monitor. If the `Monitor Type` for the monitor is `Duration`, then the threshold type is called `Relative duration`. If the `Monitor Type` for the monitor is `Process Value`, then the threshold type is called `Relative process value`. If the `Monitor Type` for the monitor is a custom defined monitor (see [Chapter 3, Custom Monitors](#)) then the threshold type is called `Relative value`.

Threshold Measure

You can set the threshold measure to be any specific instance, or a recent average/minimum/maximum.

- Any Specific Instance

This allows you to set a threshold to alarm when any monitor instance takes longer/shorter than a number of standard deviations from your overall average time. Or when the process value of any monitor instance is greater/smaller than a number of standard deviations from your overall average process value.

This threshold measure means that the moment a monitor instance hits the threshold, a violation is triggered.

- Recent

This allows you to set a threshold to alarm when the recent average (or minimum/maximum) is greater/smaller than a number of standard deviations from your overall average.

The term `recent` means the value calculated over the last collection interval of your underlying monitor. If the collection interval is 15 minutes then this threshold checks for any violations every 15 minutes.

This option is not available when the integration to Business Availability Center is enabled.

Throughput

The throughput threshold type measures the monitor instances that completed in the last collection interval for the underlying monitor. So, setting a throughput threshold type means that the value is tested at the end of each collection interval.

This threshold type is not available when the integration to Business Availability Center is enabled.

Rate

You can test the `Count` or `Process Value` of completed monitor instances expressed as a value over a given time period.

Business Availability Center Integration and Threshold Types

Once you have configured the integration to Business Availability Center, the Monitor Definer is only able to define “instance” based thresholds. That is, thresholds where you wish to trigger a violation if any particular monitor instance breaches a user-specified value. For example, trigger a violation for a monitor instance if it takes longer than two days to reach a particular step in the process. To set “statistical” thresholds, such as a threshold that measures whether a monitor instance has taken longer than the most recent average, you need to configure these thresholds within Business Availability Center.

When you configure the integration with Business Availability Center such that data samples are sent from HPBPI to Business Availability Center, the HPBPI Monitor Engine calculates the monitor values and statistics for the most recent collection interval (which is the same as the data samples send interval) and sends this set of data to Business Availability Center to be stored against the corresponding configuration item (CI) within the uCMDB. You then use the Business Availability Center Dashboard to configure your thresholds for this data.

So once you have enabled the integration between HPBPI and Business Availability Center such that data samples are being sent, you configure “instance” thresholds from within the HPBPI Monitor Definer and you configure “statistical” thresholds from within the Business Availability Center Dashboard.

Existing Thresholds

Any thresholds that you may have configured within the Monitor Definer before enabling the data sample integration with Business Availability Center, continue to work within HPBPI. In particular, if you had configured any statistical thresholds, these continue to work as before where violations are handled by the HPBPI Monitor Engine.

However, when you enable the data sample integration with Business Availability Center, all the existing monitors have their collection intervals set to the data samples send interval configured within the HPBPI Administration Console.

Warning/Minor/Major/Critical Violation

You can specify values for the different violation levels. You do not need to specify values for all violation levels, but you must provide at least one violation level value.

As a threshold crosses any specified violation value, a violation is raised within HPBPI. (See [Violations](#) on page 36 for further details.)

Violation Message

You can specify a text message to be issued when each violation is raised. This field can contain only simple text.

Violations

When thresholds are exceeded, violations are generated.

The violations are sent to two places:

- The Notification Server

The violations are sent to the Notification server and if you have configured any notification subscriptions then these are handled accordingly.

- The HPBPI Database

The violations are also written to the HPBPI database. This allows the HPBPI Business Process Dashboard to report and show violations raised.

Instance and Statistical Thresholds/Violations

As discussed in the section [Threshold Type](#) on page 31, some thresholds are measured on an “individual instance” basis and some are measured on a “collection interval” basis.

Consider setting an absolute threshold that measures when any specific monitor instance duration is greater than four hours. The moment an individual monitor instance has been running for more than four hours, a violation is issued. You can think of this threshold as an “instance threshold”, where violations are issued based on each individual monitor instance.

Now consider setting a backlog threshold that raises a violation when the backlog count is greater than 100. A backlog threshold applies to multiple monitor instances and thus is calculated at the end of each collection interval (the collection interval of the underlying monitor). So this threshold is not calculated for any individual monitor instance, instead it is calculated for all monitor instances over the last collection interval. You can think of this monitor as a “statistical threshold”, where violations are issued for the whole collection interval.

Violation Levels

Within the Monitor Definer you can choose to raise violations at the levels Warning, Minor, Major and Critical.

If the violation is based on a statistical threshold, HPBPI additionally raises a Normal violation once the measurement has dropped back below the warning level. A statistical threshold is one that is calculated for each collection interval.

Options

Within the Monitor Definer there are some options in the left-hand navigator frame.

Refreshing the Monitor Definer

If you have just deployed a new process and you want the Monitor Definer to pick up this, and any other, new definition, simply click the `Refresh` button and the Monitor Definer re-reads all deployed process definitions.

Exporting Monitors

If you have defined some monitors for a process you can export them to an external (zip) file, as follows:

1. Select the process - in the left-hand navigator frame
2. Select `File->Export->Monitors` for selected process
and save your monitor definitions, thresholds and filters to an external file name of your choice.

Exporting Monitors for All Processes

If you have monitors defined for multiple processes, you can export all monitor definitions, thresholds and filters to a single external (zip) file by selecting `File->Export->Monitors` for all processes

Collection Intervals

When the integration between HPBPI and Business Availability Center is enabled, the collection interval for all monitors is set to the value of the data sample send interval. So, when you come to export a monitor, what value does the export assign for each monitor's collection interval?

If the monitor was originally defined before you enabled the integration between HPBPI and Business Availability Center, the Monitor Definer exports the monitor and sets the collection interval to be the value that was specified when the monitor was originally defined.

If the monitor was defined after you enabled the integration between HPBPI and Business Availability Center, the Monitor Definer exports the monitor and sets the collection interval to a default value of 15 minutes.

Importing Monitors

If you have an external (zip) file that contains monitor definitions, thresholds and filters as exported from a Monitor Definer, you can import them into your Monitor Definer, as follows:

1. Select `File->Import`
2. Browse and open the desired input file
3. Click the `Import Definitions` button
4. Select the set of monitors to import
5. Click the `Import Definitions` button
6. Click `OK`

If your import file contains *custom* monitor definitions then you must have previously installed the necessary SQL stored procedures used by these custom monitors. If these stored procedures are not present within the HPBPI database at the time of the monitor import, the import fails. (See [Chapter 3, Custom Monitors](#) for more details about how to create and work with custom monitors.)

Collection Intervals

When you import a monitor, the collection interval specified on that monitor is stored within HPBPI. However, if the integration between HPBPI and Business Availability Center is enabled then the collection interval that appears within the Monitor Definer, and therefore the collection interval used for this monitor, is set to the value of the data sample send interval.

In other words, when a monitor is imported, the imported collection interval is stored with that monitor but it may not be used.

Monitor Definer Help

If you select `Help->Help Topics` the Monitor Definer help system is displayed in a separate window.

Lab - Defining Monitors

In this lab you define a new process to monitor the handling of support calls and then define a series of business process monitors. You then configure thresholds against these monitors.



This lab assumes that your HPBPI installation does not have the integration to Business Availability Center enabled. This is just so that you can create statistical thresholds and then view them from within the HPBPI Business Process Dashboard.

The Scenario

You have been brought in to monitor a call center.

In this center, customers call up with a problem and it gets logged into the system. The customer may or may not have a support contract. If the customer has a support contract, the call is then looked at by a supervisor who assigns a priority and then assigns this call to an engineer within the team. If the customer does not have a support contract then the supervisor still looks at the call and decides whether they want to take the time to answer the call. This allows the supervisor to answer calls for “potential” new customers. When a call is assigned to an engineer, they solve the problem, communicate this to the customer, and then mark the call as resolved. If a non-contract call is not assigned to an engineer, the call is simply marked closed and ignored.

This whole call system runs on a database with a front end call management application that allows the call takers, supervisors and engineers to log and update calls.

The call system is very simple. As a call comes in, it is written to the call database and the call state is set to either “Contract” or “NoContract”. As the call goes through the various stages, this state field is updated to say where it is in the process.

The call states are:

- “Contract” - this is a new call from a customer with a support contract
- “NoContract” - this is a new call from a customer with no support contract
- “Assigned” - the call is assigned to an engineer
- “Open” - the call is being worked on by the engineer
- “Resolved” - the call has been resolved
- “Closed” - the call is non-contract and to be ignored

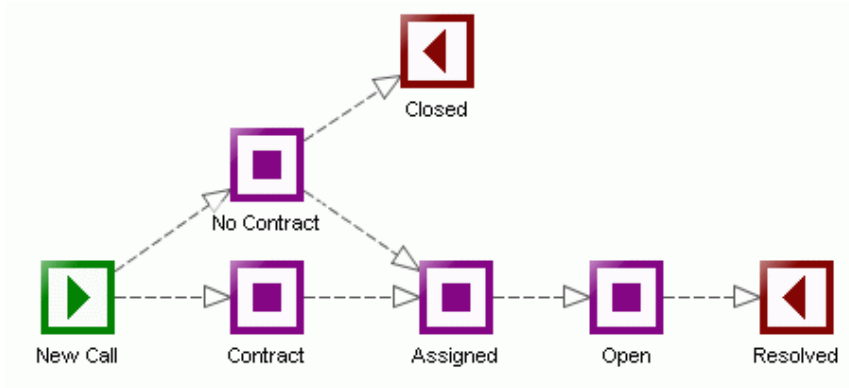
The call center would like to be able to visually see their call system on a dashboard, showing where calls are at any time. They would also like to collect the following measurements:

- The time it takes to assign contract calls
Raising a violation if the assignment time takes too long, and measuring the backlog of calls waiting to be assigned over time.
- The time it takes to process a call once it has been assigned
Raising a violation if a call takes too long to be resolved, and an indication of how fast or slow the engineers are resolving calls compared to their average resolution rate.

Defining the Process

The process definition for this lab is already defined for you. The process diagram is as shown in [Figure 10](#) on page 43.

Figure 10 Call System Process Diagram



To load up this process definition:

- Use the HPBPI Administration Console and start all Components.
- Run the HPBPI Modeler.
- Locate the file: labs\CallSystem.zip
Import this file into the HPBPI Modeler.
- Deploy the process: Call System

Understanding the Process

Now that the process definition is loaded into the HPBPI Modeler and deployed to the Business Impact Engine, let's take a few moments to understand how it works.

Data Definition

The data definition, Calls/Data, has the following data attributes:

Name	Type	Constraint	Unique
Call_ID	String	30	Yes
Customer_ID	String	30	
Call_Status	String	20	
Engineer_Name	String	50	
Call_Priority	String	2	
Call_Entry_Date	Date		

Progression Rules

The progression rules are all based around the value of the `Call_Status` attribute.

When a new call comes in, the `Call_Status` attribute is set to either `Contract` or `NoContract`. The valid states then correspond to the names of the steps.

Events

There are three event definitions, as follows:

- `Calls/New` with properties:

```
Call_ID
Customer_ID
Call_Status
Call_Entry_Date
```

- `Calls/Assigned` with properties:

```
Call_ID
Engineer_Name
Call_Priority
```

The `Call_Status` property is set to `Assigned` by the event subscription. For contract calls, the supervisor assigns a priority of either 1, 2 or 3; 1 being the highest priority. If the supervisor wants to assign and resolve a non-contract call, they assign a priority of 9.

- `Calls/Update` with properties:

```
Call_ID
Call_Status
```

Defining the Monitors

The monitors as specified by the customer are as follows:

- The time it takes to assign contract calls

Raising a violation if the assignment time takes too long, and measuring the backlog of calls waiting to be assigned over time.

- The time it takes to process a call once it has been assigned
Raising a violation if a call takes too long to be resolved, and an indication of how fast or slow the engineers are resolving calls compared to their average resolution rate.

So this can be achieved by defining two monitors and then defining a set of thresholds for these monitors.

Time to Assign Contract Calls

You need to define a monitor that measures the time spent in the `Contract` step of the process. That is, a monitor from the start of `Contract` to the end of `Contract`.

To define this...

- Start up the HPBPI Monitor Definer
- Select the `Call System` process
- Click on `Create Monitor`
- Define the monitor as follows:
 - Monitor Name: `Call Assignment Time`
 - Monitor Description: `Measure time to assign a contract call`
 - Monitor Scope: `Single Step`
 - Monitor Start Step: `Contract`
 - Monitor Type: `Duration`
 - Statistics Collection: `On`
 - Collection Interval: `5 minutes`
 - Apply Filter: `None`
 - Group Results By: `None`
 - Deadline Property: `None`
- Click `OK`

You have now defined a monitor that measures the time that contract calls spend waiting to be assigned.

In the real world you would probably choose the default collection interval or a larger collection interval, however, in this lab situation you are choosing a smaller collection interval just so you have quicker results within the lab time of the class.

Raise a Violation if Duration Too Long

You now need to define a threshold for this monitor to raise a violation if the time it takes to assign any individual call takes too long. In the real world the term “too long” might be four hours, or one day, but for this lab let’s make that a lot smaller. Let’s make the limit six minutes:

- Within the Monitor Definer, select the `Call Assignment Time` monitor
- Click `Create Threshold`
- Create a threshold as follows:
 - `Threshold Name`: `Call Assignment SLA`
 - `Threshold Type`: `Absolute duration`
 - `Threshold Measure`: `Any specific instance duration`
 - `Duration Units`: `Minutes`
 - `Test Condition`: `Greater than or equal to`
 - `Critical Violation`: `6`
- Click `OK`

Call Assignment Backlog

The customer also wants you to monitor the backlog of calls waiting to be assigned:

- Select the `Call Assignment Time` monitor
- Click `Create Threshold`

- Create a threshold as follows:
 - Threshold Name: Call Assignment Backlog
 - Threshold Type: Backlog
 - Threshold Measure: Count
 - Test Condition: Greater than or equal to
 - Major Violation: 15
 - Critical Violation: 25
- Click OK

Time to Process Assigned Calls

To define a monitor to measure the time taken to resolve a call once it has been assigned, seems straight forward. You just need to define a monitor with a multiple step scope that measures from the start of the `Assigned` step to the end of the `Resolved` step. However, there are some additional considerations:

- The customer only wants to measure this for actual contract calls. So you want a filter that filters this monitor such that only contract calls (calls with a priority of 1, 2 or 3) are included.
- When a support call is placed, the call comes in with a `Call_Entry_Date`. The expectation that the customer has is that all contract calls are handled within a certain time from the time the call is entered into the system. That is, the call resolution time has a deadline based on the `Call_Entry_Date`.
- You decide that the customer might like to be able to view the processed calls information grouped by call priority. This would enable them to view the day's calls and see not just the volume of calls, but a breakdown of calls by priority.

So, let's define the monitor...

But first...you must define the filter:

- Select the `Call System` process (in the left-hand navigator pane)
- Select the `Filters` tab
- Click `Create Filter`

- Set the filter name to be: Call On Contract
- In the Data Definition text box click on the Call_Priority attribute and then click on the Paste button

This pastes the full name of this data attribute into the Filter Expression text box below.

- In the Filter Expression text box, complete the expression so that it ends up looking as follows:

```
data.Call_Priority.in("1", "2", "3")
```

- Click OK

Now to define the monitor:

- Select the Call System process
- Click Create Monitor
- Define the monitor as follows:
 - Monitor Name: Call Processing Time
 - Monitor Scope: Multiple steps
 - Monitor Start Step: Assigned (start)
 - Monitor End Step: Resolved (complete)
 - Monitor Type: Duration
 - Statistics Collection: On
 - Collection Interval: 5 minutes
 - Apply Filter: Call On Contract
 - Group Results By: Call_Priority
 - Group Name: Call Priority
 - Deadline Property: Call_Entry_Date
- Click OK

Again, because this is a lab scenario and you want to see results more quickly, you are configuring a small collection interval.

Now to define the thresholds...

Raise a Violation if Deadline Not Met

The support agreement is that contract calls must be resolved within a certain time from when they were entered. Again, being a lab scenario, let's make this time short so you can test it out without having to wait too long.

Let's configure a threshold to issue a critical violation if the time to process a contract call is 35 minutes from the time the call was entered. Let's also issue an additional minor violation if the call reaches 20 minutes from the time the call was entered:

- Select the **Call Processing Time monitor**
- Click **Create Threshold**
- Create a threshold as follows:
 - **Threshold Name:** Call Time SLA
 - **Threshold Type:** Deadline
 - **Minor Violation:** 20 Minutes After
 - **Critical Violation:** 35 Minutes After
- Click **OK**

Call Processing Speed

The customer also requested to monitor how fast or slow the engineers are resolving calls compared to their average resolution rate:

- Select the Call Processing Time monitor
- Click Create Threshold
- Create a threshold as follows:
 - Threshold Name: Call Processing Speed
 - Threshold Type: Relative duration
 - Threshold Measure: Recent Average duration
 - Test Condition: Less than or greater than usual
 - Warning Violation: 1.0
 - Minor Violation: 1.5
 - Major Violation: 2.0
 - Critical Violation: 2.5
- Click OK

Great! You should have the following monitors and thresholds defined for your Call System process:

Monitor: Call Assignment Time (Single step, duration)

Threshold: Call Assignment Backlog (Backlog)
Threshold: Call Assignment SLA (Absolute)

Monitor: Call Processing Time (Multiple step, duration)

Threshold: Call Processing Speed (Relative)
Threshold: Call Time SLA (Deadline)

Now that these monitors are defined within the Monitor Definer, they are active. Indeed, your monitor collection intervals may have already kicked into action and data may have already started to collect in the HPBPI database. Of course, the statistics at the moment simply record lots of zeros because there are no calls moving through your system. So how are you going to get loads of calls into the system so that you can track and measure their performance? The answer is the **Process Simulator...**

The Process Simulator

The Process Simulator is a **contributed utility** that allows you to inject events into HPBPI. The big benefit of the Process Simulator is that you can store these events as `Test Cases`, and then run these test cases as and when you need. The Process Simulator can drive any process, and has some pre-defined tokens that allows you to substitute special values such as unique IDs and date/times.

The Process Simulator is located under the `HPBPI-install-dir\contrib\ProcessSimulator` directory.

To run the Process Simulator, you just need to double-click the `ProcessSimulator.bat` file.

If you want to learn more about how to use the Process Simulator, please refer to the PDF documentation that is provided in the `contrib\ProcessSimulator` directory.

Running the Call Center

For this lab, you are provided with a pre-defined set of test cases to run within the Process Simulator. These test cases inject contract and non-contract calls through the `Call System` process, allowing you to run the HPBPI Business Process Dashboard and see the calls and the monitors.

Let's load up the Process Simulator:

- `cd` to the directory: `HPBPI-install-dir\contrib\ProcessSimulator`
- Run (Double-Click) the script: `ProcessSimulator.bat`
This runs the Process Simulator in a separate GUI
- In the Process Simulator GUI, select: `File->Open Test Suite`
- Open the file: `labs\Calls.xml`

The Process Simulator should load this file and you should now see a number of test cases shown in the top-left-hand corner of the Process Simulator window.

Test Cases

The test cases have names that describe what they do. Let's go through them:

- NoContract-Resolved

This set of events send through a call that is from a non-contract customer. The call is assigned to an engineer and resolved.

- NoContract-Closed

This set of events send through a call that is from a non-contract customer. The call is not-assigned to an engineer and simply closed.

- Contract-Pri-1

This set of events send through a call that is from a contract customer. The call is assigned to an engineer with a priority of "1", and resolved.

- Contract-Pri-2

This set of events send through a call that is from a contract customer. The call is assigned to an engineer with a priority of "2", and resolved.

- Contract-Pri-3

This set of events send through a call that is from a contract customer. The call is assigned to an engineer with a priority of "3", and resolved.

Running the Suite of Tests

Now that you have loaded the Process Simulator with a set of test cases, let's run them:

- Within the Process Simulator GUI, click on the `Test Suite Runner` tab

This displays the individual test cases with `Start` and `Stop` buttons for each. It also displays a slider control to adjust how often each test case is run.

Each test case is made up of a set of events. Below the `Event Injectors` heading you see a start/stop control and a slider control for each event.

All the sliders are pre-set to initial values appropriate for this lab.

- Click the `Start Suite` button at the bottom of the screen and this starts the test suite.

The sliders are pre-set such that calls take a while to be assigned. You should start to notice a build-up of calls waiting to be assigned.

Running the HPBPI Business Process Dashboard

- With the Process Simulator still running, start up the HPBPI Business Process Dashboard and drill into the `Call System` process.

You should start to see a gradual build up of calls in the `Contract` step.

Notice that as well as seeing the process diagram for the `Call System` process you also see dials and tables representing the four thresholds that you defined. You may not see values in the dials until the first collection intervals have occurred. Remember that statistical thresholds are only updated after each collection interval.

The HPBPI Business Process Dashboard displays the following graphics for defined thresholds:

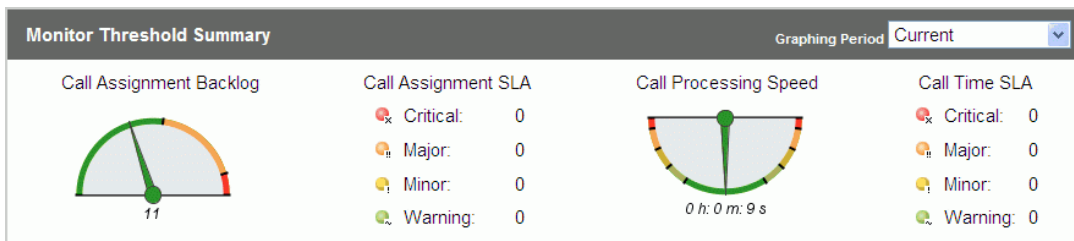
- A dial for each statistical thresholds (excluding `Relative` thresholds)
- An upside-down dial for each `Relative` statistical threshold
- A table of instance violations for each instance threshold

These graphics help you see the overall state of your business at a glance.

- Let the Process Simulator run for five minute...
- After the first collection interval is complete you should see values displayed in the dials for your monitors.

For example, the threshold dials and violation tables might look as shown in [Figure 11](#).

Figure 11 Monitor Threshold Summary Table



Call Processing Speed Dial

The Call Processing Speed dial is shown as an upside-down dial, called a “swing dial”, because it is showing the relative movement towards one side or the other. That is, the needle represents the most recent average time taken to process calls. Remember that the most recent average is the average over the most recent collection interval, which is five minutes in this case. The needle points straight down (to the center of the green area) if the most recent average is the same as the overall average (the average since the monitor was first defined). If the most recent average is faster than the overall average then the needle swings to the right. If the most recent average is slower than the overall average, then the needle swings back towards the left side.

So the swing dial is used to represent relative thresholds, and the needle position indicates the relative position towards “faster than the average” or “slower than the average”.

The value shown under the swing dial is the most recent average value. If this value is greater than the overall average than the needle swings to the right, other wise the needle swings to the left.

The swing dial allows you to visually see whether your call processing time is getting slower (a swing to the right) or getting faster (a swing to the left).

More Graphs

- Click on the Call Assignment Backlog dial and/or the Call Processing Speed dial to see historical graphing of the values

This may not be very exciting until your Call Center events have been running through the system for a few collection intervals.

You can also see a number of different types of graphs for the underlying monitors:

- In the HPBPI Business Process Dashboard, on the main Business Process & Resource Summary page - the page that shows the process diagram and the threshold dials - click on the tab marked: Monitors
- Click on the monitor called: Call Processing Time

By default this shows you a graph displaying the Average/Minimum/Maximum values for each collection interval.

- Select the `Data Source` for the graph to be: `Completed - Count`
This shows you the number of calls that have been resolved within each collection interval.
- Now select `Chart by group` to be: `Yes`
You now see the number of calls that have been closed within each collection interval, but now showing the numbers for each call priority.

Many of these graphs may not show much data as you have only just started collecting measurements. At the end of this lab you should leave the Process Simulator running and thus, collect more data that you can drill into during the next lab.

Adjusting the Process Simulator

- Once the HPBPI Business Process Dashboard is showing violations for the `Call Assignment SLA` threshold you need to go back to the Process Simulator and speed up the time it takes for calls to be assigned.
- In the Process Simulator, move the slider for the event `Calls/Assigned` to the left. You should move the slider until the number to the right of the slider is showing (about) 5000ms.

This tells the Process Simulator to handle these events at one event every 5000 milliseconds (five seconds) and thus the calls currently in the `Contract` and `No Contact` steps are processed more quickly.

Back in the HPBPI Business Process Dashboard, you should notice that calls waiting in the `Contract` step start to reduce.

- After the next collection interval, notice how the dials indicate that there is less of a call assignment backlog and that orders are starting to take a little longer to be processed (the `Call Processing Speed` dial swings to the right).

End of The Lab

- At this point, you have finished the lab! However, you should **leave the Process Simulator running**.
- Over time, make periodic adjustments to the Process Simulator sliders to simulate changes in the speed at which calls are being handled.

As you make any changes, leave them in place for about 10 minutes before changing them again.

For example:

- Move the `Calls/Update` slider down (left) to a very small time value
This simulates calls being resolved very quickly.
 - Move the `Calls/Update` slider up (right) to a large value
This simulates calls being resolved very slowly.
 - Move the `Contract-Pri-1` slider all the way to the right
This simulates few priority-1 calls coming into the system
- You can view the monitor data in the HPBPI Business Process Dashboard and see the affects of your changes.

Well done! You have reached the end of the lab.

2 Monitor Engine

The central Business Impact Engine processes all incoming events and progresses all process instances. Rather than increase the workload of the Business Impact Engine by asking it to also calculate all business monitors, it was decided to create a separate HPBPI component to handle monitors. This component is called the **Monitor Engine**.

This chapter looks at the Monitor Engine and provides an overview of the underlying SQL data tables that are used to hold monitor data.

How Monitors Work

Monitor Events

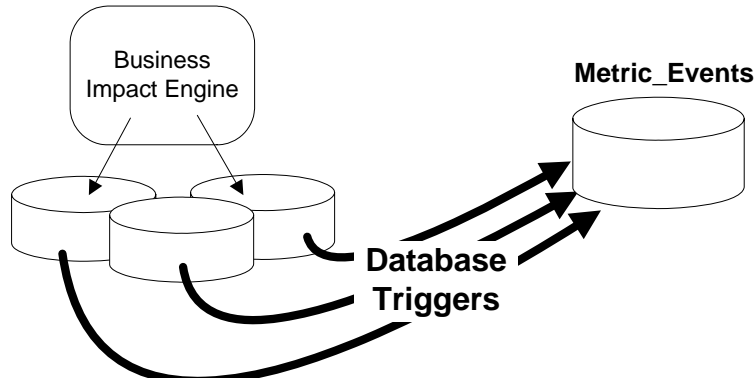
As process and step instances are created and updated, the central Business Impact Engine updates the tables within the HPBPI database. By placing triggers on some of the key tables, as a process instance enters a step that starts a monitor, a new monitor instance is created. The same is true when a process instance triggers the fact that a monitor has completed.

These monitor records are written into the data table `Metric_Events`, within the HPBPI database, as and when they occur.

▶ In previous versions of HPBPI, monitors used to be referred to as “metrics”. Hence many of the database table names still use to the term “metric”.

The creation of monitor records within the `Metric_Events` table is shown in [Figure 12](#).

Figure 12 Creation of Monitor Event Records



where:

- The Business Impact Engine updates the database tables as normal.
- SQL database triggers on the various data tables capture the details whenever a monitor starts or completes.
- The necessary monitor data is written to the `Metric_Events` data table.

There are four types of monitor record written to the `Metric_Events` table:

- `Started`
Records the starting of a monitor instance.
- `Completed`
Records the completion of a monitor instance
- `EndFlow`
Records the end of a process instance. If a monitor instance is still active, even though its underlying process instance has now completed, the Monitor Engine marks the monitor instance as `Aborted`.
- `NewWeight`
Records the process value for a monitor instance. This record is written when the process value changes during the life of a monitor instance. (The process value used to be called the process “weight”.)

Each record written to the `Metric_Events` table contains all the information needed for the recording of each monitor.

The `Metric_Events` table is the input for the HPBPI Monitor Engine.

Monitor Values

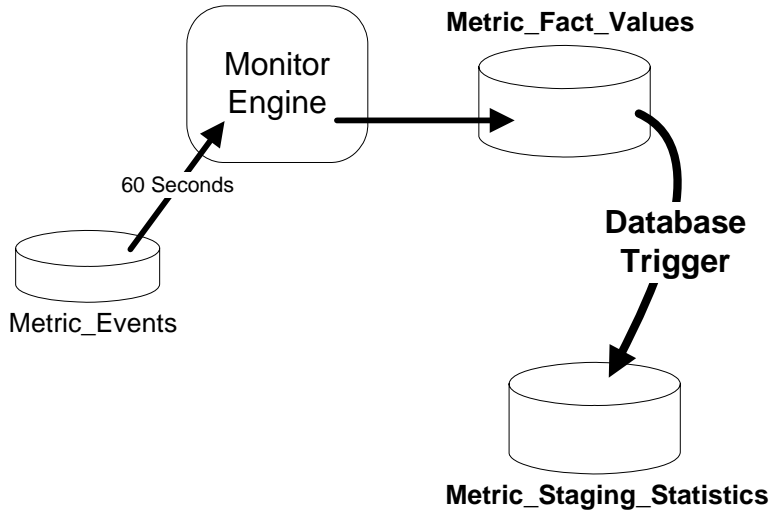
Once the monitor record is written into the `Metric_Events` table it is up to the Monitor Engine to process this record.

The first stage of the process is to record all the individual monitor values. This is where it matches up the start and complete records from the `Metric_Events` table and record the actual duration and/or process value for each monitor instance. Each monitor instance record is written to the `Metric_Fact_Values` table.

There is a database trigger on the `Metric_Fact_Values` table. As monitor instances are created and updated within the `Metric_Fact_Values` table, this trigger creates entries within another table, called `Metric_Staging_Statistics`. The `Metric_Staging_Statistics` table is used as a staging area, or work area, for the future calculation of the monitor statistics after each collection interval. You do not really need to know (or worry) about the `Metric_Staging_Statistics` data table as it is just a temporary work area for the Monitor Engine. It is just mentioned here because it plays a part in the production of monitor statistics.

Monitor values are generated as shown in [Figure 13](#).

Figure 13 Monitor Values



where:

- The Monitor Engine processes the records from the `Metric_Events` table.

When there are records in the `Metric_Events` table, the Monitor Engine continues to process all records. When there are no more records in the `Metric_Events` table the Monitor Engine sleeps for the configured interval (60 seconds by default).

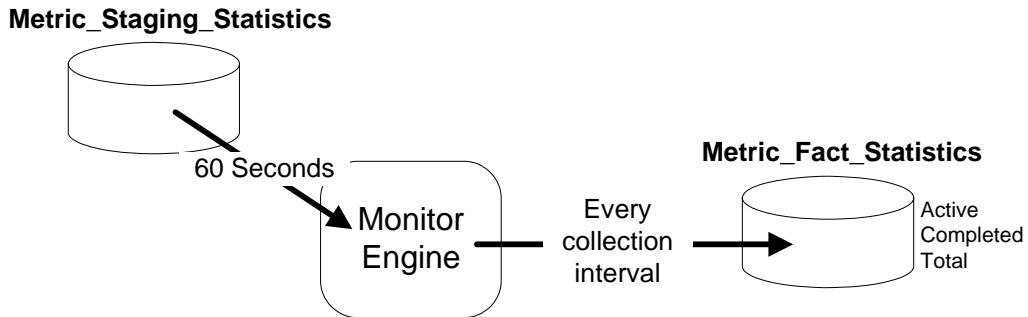
This sleep period is called `The monitor events idle wait time` and is configurable through the HPBPI Administration Console. The default setting is 60 seconds.

- The `Metric_Fact_Values` table holds one record for each monitor instance.
- The `Metric_Staging_Statistics` table holds intermediate data that enables the Monitor Engine to calculate statistics when each monitor collection interval occurs.

Monitor Statistics

Monitor statistics are produced at the end of each collection interval. These statistics are generated from the data collected within the `Metric_Staging_Statistics` data table. Let's draw the picture and then explain this in more details...

Figure 14 Monitor Statistics



where:

- The Monitor Engine is polling the `Metric_Staging_Statistics` table every 60 seconds.

This poll period is called the `Statistical generation polling interval` and is configurable through the `HPBPI Administration Console`. The default setting is 60 seconds.

- If a collection interval has occurred, the Monitor Engine writes three summary records into the `Metric_Fact_Statistics` table.

The Monitor Engine produces three records to summarize that collection interval:

- Active

The records record details for the monitor instances that are still active.

- Completed

The monitor instances that have completed within that collection interval

— Total

The overall total for all monitor instances completed over all collection intervals so far.



So the `Metric_Fact_Statistics` table is probably the table that can grow the fastest. Every collection interval, three records are being written to this table for every monitor defined on your system. These three records are written every collection interval even if nothing else has happened on your system.

Group Results By

If you have configured a monitor to have a `Group Results By` property, then the Monitor Engine writes more than three summary records at each collection interval.

When a monitor is defined to have a `Group Results By` property, the Monitor Engine writes out the following records at the end of each collection interval:

- A set of three records (`Active/Completed/Total`) for each current group within the monitor.
- A set of three records (`Active/Completed/Total`) for the monitor overall.

For example, suppose you are collecting monitors for an airport and grouping the monitor data by each airport terminal. If there are four airport terminals, then at each collection interval the Monitor Engine is writing 15 records to the `Metric_Fact_Statistics` table. These 15 records consist of three records for each terminal, and three records for the overall statistics.

Monitor Violations

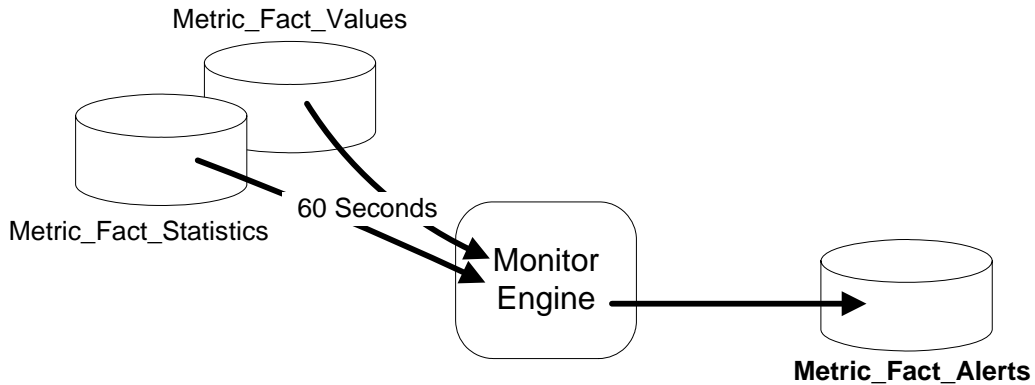
With monitor values being recorded in the `Metric_Fact_Values` table as they happen, and monitor statistics being calculated and stored in the `Metric_Fact_Statistics` table every collection interval, the Monitor Engine can monitor these two tables to see when any thresholds have been exceeded.

As a threshold is exceeded, a violation can be raised.

- ▶ In previous versions of HPBPI, violations used to be referred to as “alerts”. Hence many of the database table names still use to the term “alert”.

The handling of violations looks as shown in [Figure 15](#).

Figure 15 Monitor Violations



where:

- The Monitor Engine polls the `Metric_Fact_Values` and the `Metric_Fact_Statistics` tables every 60 seconds looking to see if any thresholds have been exceeded.

This poll period is called the `Threshold polling interval` and is configurable through the HPBPI Administration Console. The default setting is 60 seconds.

Instance thresholds are measured against the monitor values held in the `Metric_Fact_Values` table. Statistical thresholds are measured against the statistics collected in the `Metric_Fact_Statistics` table.

- If a threshold has been exceeded the Monitor Engine writes the violation into the `Metric_Fact_Alerts` data table. These violations are then visible from within the Business Process Dashboard.

Monitor Notifications

With the `Metric_Fact_Alerts` table holding all the violations, the Monitor Engine can monitor this table and send these violations to the HPBPI Notification Server as Process Monitor Threshold Violation notifications; see [Figure 16](#) on page 65.

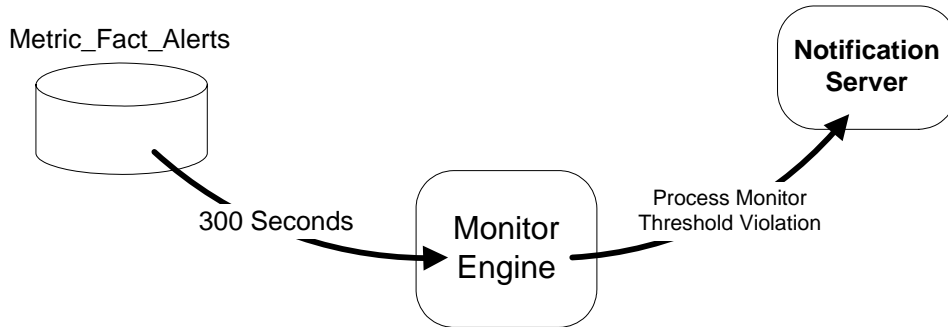
The Monitor Engine polls the `Metric_Fact_Alerts` table every 300 seconds. This poll period is called the Threshold violation notification polling interval and is configurable through the HPBPI Administration Console.

You can also configure the maximum number of violation notifications that are to be sent to the HPBPI Notification Server within any single polling period, and this maximum is applied separately for each threshold. This maximum is set to 10 by default, but you can configure this using the HPBPI Administration Console. The idea of setting a maximum number of notifications for each poll period is to help guard against creating and sending too many notifications at each polling interval.

For example, suppose you have two thresholds - `Thresh-A` and `Thresh-B`. Let's suppose that, during one polling interval (of 300 seconds) 100 monitor instances breach `Thresh-A` and 50 monitor instances breach `Thresh-B`. Rather than send out 150 notifications, the Monitor Engine sends out the most recent 10 notifications for `Thresh-A` and the most recent 10 notifications for `Thresh-B`. All the violations (all 150 of them) remain in the `Metric_Fact_Alerts` table, but only 20 notifications are sent to the HPBPI Notification Server.

Each notification message sent to the HPBPI Notification Server contains the details of the individual notification, as well as a summary table that shows how many violations occurred during the 300 seconds polling period. This summary table shows the violations grouped by severity. Each notification sent for this polling period contains the same summary table.

Figure 16 Monitor Notifications



where:

- The Monitor Engine polls the `Metric_Fact_Alerts` table every 300 seconds looking to see any new violations.

This poll period is called the `Threshold violation notification polling interval` and is configurable through the `HPBPI Administration Console`. The default setting is 300 seconds.

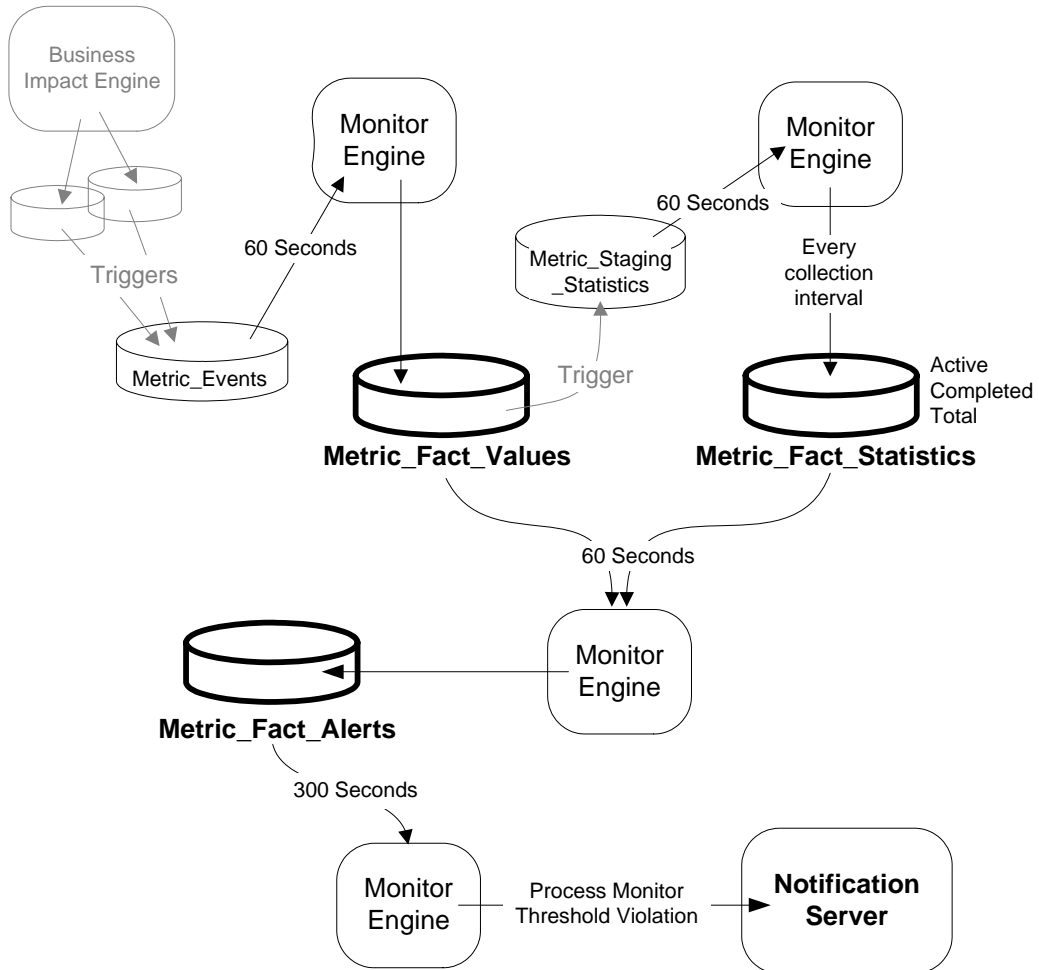
- The violations are sent to the `HPBPI Notification Server` as `Process Monitor Threshold Violation` notifications.

The overall number of notifications sent at each polling interval can be limited. This is configurable from within the `HPBPI Administration Console`.

The Big Picture

Putting all the diagrams together that describe how monitor values, statistics and violations are processed, gives you the overview diagram shown in [Figure 17](#).

Figure 17 Monitors Overview



Violation Timings

When you have configured a threshold you may ask yourself the question: “Exactly when is the violation going to be raised?”

If you look at [Figure 17](#) on page 66 you see that there are some small delays when the Monitor Engine is polling the various data tables. This can mean that the violations may be raised a few seconds/minutes after the threshold has actually been exceeded. And of course, if the threshold is based on a statistical monitor then there is also the collection interval to be considered.

Don’t forget that the monitor polling periods can all be configured using the HPBPI Administration Console.

Let’s look at the maximum delays that may occur when raising violations, and using the default monitor polling periods.

Instance Thresholds

Let’s consider thresholds that are set against instances. For example, a threshold that tests an instance taking longer than four hours to get to a certain step within a process.

There is up to a 60 second delay before monitor values are written to the `Metric_Fact_Values` table.

There is up to a 60 second delay before the `Metric_Fact_Values` table is polled to look for instance threshold violations.

So there is up to a 2 minute delay before a violation is raised when an instance exceeds a threshold.

The actual delay is less than or equal to 2 minutes because it depends when the violation occurs. For example, the violation might be written to the `Metric_Fact_Values` table one second before the next poll of that table, so the total delay in spotting the violation might be only a few seconds.

When a violation is written to the `Metric_Fact_Alerts` data table, it records both the time the violation occurred and the time the violation was actually spotted by the Monitor Engine.

There is then a further delay of up to 300 second before any notification violation is sent to the HPBPI Notification Server.

Statistical Thresholds

When the threshold is set against a statistical value (such as an average value, or a backlog), the potential delays are as follows:

There is the actual collection interval. If the collection interval is set to 30 minutes then the statistical value is not calculated until that 30 minutes has passed.

There is up to a 60 second delay before monitor values are written to the `Metric_Fact_Values` table.

There is up to a 60 second delay before the `Metric_Staging_Statistics` table is polled to see if a collection interval has occurred. Then the statistical results are written to the `Metric_Fact_Statistics` table.

There is up to a 60 second delay before the `Metric_Fact_Statistics` table is polled to look for instance threshold violations.

So the total, potential delay in raising a violation for a statistical threshold is:

`60 seconds + 60 seconds + 60 seconds + Collection Interval`

When a violation is written to the `Metric_Fact_Alerts` data table, it records both the time the violation occurred and the time the violation was actually spotted by the Monitor Engine.

There is then a further delay of up to 300 second before any notification violation is sent to the HPBPI Notification Server.

Monitor Engine Off/Restart

If you look at [Figure 17](#) on page 66 you realize that you can turn off the Monitor Engine and not lose any monitor data. With the Monitor Engine turned off, the monitor events simply accumulate in the `Metric_Events` table.

When the Monitor Engine is restarted, after being off for a period of time, the `Metric_Events` table is processed and the monitor values are recorded in to the `Metric_Fact_Values` table in the normal way, and the monitor statistics recorded in to the `Metric_Fact_Statistics` table in the normal way.

Violations

While the Monitor Engine is turned off, no violations are going to be raised. However, when the Monitor Engine is restarted the monitor data is processed and violations may well be raised. The violations are raised showing both the time the violation actually occurred on your system and the time the Monitor Engine actually spotted the violation.

Statistical Monitors ("Back Filling")

When the Monitor Engine is restarted, after being off for a period of time, it must calculate the statistics for each monitor for each collection interval for all the time the Monitor Engine was switched off.

That is, if you have a monitor with a collection interval set to (for example) five minutes and you have turned the Monitor Engine off for four hours, when you restart the Monitor Engine it goes through and calculates statistic records for all the five minute boundaries across the last four hours and writes these into the `Metric_Fact_Statistics` table. In other words, the Monitor Engine “back fills” the statistics so that they appear as if the Monitor Engine has never been turned off.

This “back filling” of statistical data is fine if the Monitor Engine has just been off for a few minutes or hours. In these situations you want the Monitor Engine to keep all the monitor statistics up to date. But what if you have turned off the Monitor Engine for a week? Do you really want the Monitor Engine to go through and “back fill” all the statistics for the last week? Possibly not.

The Monitor Engine has a user configurable setting that specifies how far back to calculate statistics after a restart. This setting is called:

Maximum age of generated statistics on startup

and it is measured in days.

The default setting is one day.

So, by default, if you turn the Monitor Engine off for anything less than a day the Monitor Engine calculates all the statistics for the time it was turned off. But if you turn the Monitor Engine off for longer than a day, statistics are only calculated back for one day's worth.



“Back filling” can have a significant impact, particularly in cases where you are preparing for a demonstration. As an example, if you set up your system one day for a demonstration on the next day, you need to make sure that you allow time for the Monitor Engine to complete its “back filling” before starting your demonstration. If you do not, it might take some time before any new monitor data is available to be viewed through the Business Process Dashboard during your demonstration session.

Instance Cleaner Settings

Monitor Instance Cleaner

In the HPBPI Administration Console you can configure the Monitor Instance Cleaner to remove monitor values (Active and Completed), monitor statistics and monitor violations. You can configure to remove some or all of these record types and you can configure the age of the data to be removed.

Over time, the number of monitor instances and statistics can grow quite large, so you need to make sure that your HPBPI database, and your server's disc space, is large enough for the amount of monitor data you need to have available. By configuring the Monitor Instance Cleaner you can help prevent the monitor data from growing too large.

Business Impact Engine Instance Cleaner

The Business Impact Engine has its own Instance Cleaner which can be configured to remove old process and data instances from the HPBPI database.

If the Business Impact Engine Instance Cleaner removes a process instance for which there is monitor data, that is ok. When the monitor record is written to the `Metric_Events` data table, it contains all the data required to record the monitor. The Monitor Engine does not require any other process instance or data instance data to calculate the monitor statistics.

Mind you, within the HPBPI Business Process Dashboard, when displaying monitor details, if a monitor links to a process instance, the Business Process Dashboard wants to show this monitor data with a live link to the related process instance data. If at the time of display the related process instance has been removed by the Business Impact Engine Instance Cleaner, the Business Process Dashboard simply shows the monitor data without an active link to the process instance.

3 Custom Monitors

What if the HPBPI monitors do not give you the exact monitor that you wish to measure? Maybe you need to measure the value of your orders however the value you need is not held in the process value property? Maybe you need to measure the percentage of orders that have gone through a particular step?

If the monitor you need to record is not available using the standard out-of-the-box monitors, you can add your own.

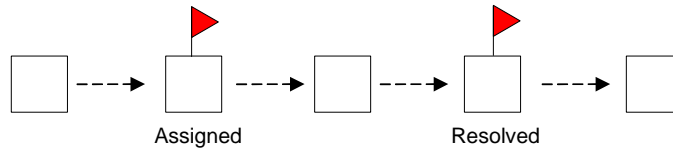
This chapter looks at how to create and use your own custom monitors.

Monitor Scope

The scope of the monitor determines when the start of each monitor instance is recorded and when the end of the monitor instance is recorded.

For example, suppose you configure a monitor, and set the monitor scope to be from the start of the step `Assigned` to the end of the step `Resolved`:

Figure 18 Monitor Scope



When a process instance enters the `Assigned` step, a new monitor instance is started. When this process instance leaves the `Resolved` step, the monitor instance is completed.

In [How Monitors Work](#) on page 58 you learnt that the start and end of the monitor instance is captured by triggers on the Business Impact Engine tables. These triggers capture the starting monitor value and write this to the HPBPI database. The triggers also capture the final monitor value at completion and write this to the HPBPI database.

The triggers that capture the start and end of a monitor instance, call predefined SQL stored procedures to capture the monitor value. You are able to provide your own custom stored procedures and have these called when a monitor starts and completes. Thus you can write a custom stored procedure to capture whatever value you are required to measure.

Defining a Custom Monitor

There are two main parts to defining a custom monitor:

1. The stored procedure

Writing the stored procedure to determine the actual monitor value.

2. The Monitor Definer

Making this stored procedure available within the Monitor Definer such that users can define monitors that use this stored procedure.

The Stored Procedure

For your stored procedure to be called correctly it must accept the following parameters:

```
Monitor Id
Process Id
Process Instance Id
Process Instance Identifier
Process Instance start time
Process Instance start time in milliseconds
Process Value
Data Definition ID
Data Instance ID
Event type
Event time
Event time in milliseconds
Index
```

Parameter Values

Let's consider the values to be contained in these input parameters:

- Monitor Id

This is the database unique ID for the monitor.

- Process Id
Process Instance Id
Process Instance Identifier

Process Instance start time
Process Instance start time in milliseconds
Process Value

These are the IDs and values for the process instance for which this monitor value is to be determined.

- Data Definition ID

This is the unique ID for the process's related data table entry in the `Data_Objects` table.

- Data Instance ID

This is the unique ID of this process's related data instance entry within the related data table.

- Event type

This is a string attribute that contains one of two values:

— Started

This is the start of a new monitor instance.

— Completed

This is the completion of the monitor instance.

- Event time
Event time in milliseconds

The time that this monitor is being recorded.

- Index

It may be that a process instance passes through the step that starts the monitor, more than once. You can use the index value to code for this.

Zero (0) means that it is the first time through this step. One (1) means this is the second time through this step, etc.

SQL Parameter Syntax

As the SQL syntax for MSSQL and Oracle is different when defining stored procedures, here are two examples for how to define the parameters:

For MSSQL:

@Monitor_ID	NVARCHAR (36) ,
@Process_ID	NVARCHAR (36) ,
@ProcessInstance_ID	NVARCHAR (36) ,
@ProcessInstIdentifier	NVARCHAR (40) ,
@ProcessInstStartTime	DATETIME,
@ProcessInstStartTimeLongMillis	BIGINT,
@ProcessValue	FLOAT,
@DataDefinition_ID	NVARCHAR (36) ,
@DataInstance_ID	NVARCHAR (36) ,
@EventType	NVARCHAR (12) ,
@EventTime	DATETIME,
@EventTimeLongMillis	BIGINT,
@Idx	INTEGER

For Oracle:

Monitor_ID	VARCHAR2 ,
Process_ID	VARCHAR2 ,
ProcessInstance_ID	VARCHAR2 ,
ProcessInstIdentifier	VARCHAR2 ,
ProcessInstStartTime	DATE,
ProcessInstStartTimeLongMillis	NUMBER,
ProcessValue	FLOAT,
DataDefinition_ID	VARCHAR2 ,
DataInstance_ID	VARCHAR2 ,
EventType	VARCHAR2 ,
EventTime	DATE,
EventTimeLongMillis	NUMBER,
Idx	NUMBER

The Monitor Value

Once your stored procedure has determined the monitor value, it needs to write this value into the `Metric_Events` data table. To do this, there is a pre-defined stored procedure called `METRIC_SEND_EVENT_V2`. (The `_V2` at the

end of the procedure name stands for “version 2”. Refer to [Upgrading Custom Monitors](#) on page 115 for discussions about converting old style custom monitors to use version 2 of this procedure.)

The `METRIC_SEND_EVENT_V2` stored procedure accepts the following parameters:

```
Monitor Id
Process Id
Process Instance Id
Process Instance Identifier
Process Instance start time
Process Instance start time in milliseconds
Process Value
Data Definition ID
Data Instance ID
Event type
Event time
Event time in milliseconds
Monitor Value
Index
```

These are the same parameters as were passed into your stored procedure, with the addition of the `Monitor Value` parameter.

As the SQL syntax for MSSQL and Oracle is different when calling stored procedures, here are two examples for how to define the call to the `METRIC_SEND_EVENT_V2` stored procedure:

For MSSQL:

```
EXECUTE METRIC_SEND_EVENT_V2 @Monitor_ID,
                              @Process_ID,
                              @ProcessInstance_ID,
                              @ProcessInstIdentifier,
                              @ProcessInstStartTime,
                              @ProcessInstStartTimeLongMillis,
                              @ProcessValue,
                              @DataDefinition_ID,
                              @DataInstance_ID,
                              @EventType,
                              @EventTime,
                              @EventTimeLongMillis,
                              @Value,
                              @Idx
```

For Oracle:

```
METRIC_SEND_EVENT_V2 (Monitor_ID,  
    Process_ID,  
    ProcessInstance_ID,  
    ProcessInstIdentifier,  
    ProcessInstStartTime,  
    ProcessInstStartTimeLongMillis,  
    ProcessValue,  
    DataDefinition_ID,  
    DataInstance_ID,  
    EventType,  
    EventTime,  
    EventTimeLongMillis,  
    Value,  
    Idx);
```

Monitor Backlog

Your stored procedure is called both at the start of the monitor and the completion of the monitor. Indeed, your stored procedure is able to test whether it is being called for the start or completion of a monitor by testing the `EventType` parameter.

Suppose the custom monitor that you are measuring always has a zero start value. In this case you might decide that you do not need to write any monitor record at the start of the monitor, and only write a monitor record at the completion of the monitor. That is, only calculate the monitor value and call `METRIC_SEND_EVENT_V2` if the `EventType` is `Completed`.



Only writing completing monitor records cuts down on some monitor processing time and works just fine. However, if you choose to not write any monitor record at the start of the monitor, you are not able to observe and graph the backlog for your monitor. Without any start-of-monitor records there is no way to show how many monitor instances are currently active and hence no way to represent the monitor backlog. So if you want to be able to monitor your custom monitors, including the backlog, then call `METRIC_SEND_EVENT_V2` for **all** invocations of your stored procedure.

Stored Procedure Ownership

When you create your custom monitor stored procedure, you need to ensure that your stored procedure is owned by the same user that owns the HPBPI data tables. The default user name for this is `hpbpiuser`, however this name is configurable at HPBPI installation time.

If your custom monitor stored procedure is not owned by the same user as the HPBPI data tables, the HPBPI Business Impact Engine may not be able to invoke it, and this may cause the HPBPI Business Impact Engine to log errors and be unable to execute.

The Monitor Definer

Once you have defined and installed your stored procedure to capture and record your monitor values, you need to make this available to the Monitor Definer so that users can associate this stored procedure with a monitor definition.

The Monitor Definer looks in the table `METRIC_CustomTypes` to locate any custom monitor types that are defined.

Defining a Custom Type

You need to add a record to the `METRIC_CustomTypes` table to describe your new custom monitor and connect it to your stored procedure.

The `METRIC_CustomTypes` table has the following columns:

- `CustomMetricName`

You specify the name for your custom monitor. This name appears in the `Monitor Type` attribute in the Monitor Definer when defining a monitor. The user are able to select this monitor from the pull-down list.

- `CustomMetricDescription`

You provide a description for your custom monitor. This is purely for your own documentation purposes.

- `CustomSPName`

This is where you specify the name of your custom stored procedure.

- ValueUnits
You can specify the display name to be used within the HPBPI Business Process Dashboard when displaying the values of this custom monitor.
- ParameterVersion
This needs to be set to the number 2.

Here is an example SQL command to create a new monitor custom type definition:

```
INSERT INTO METRIC_CustomTypes
    (CustomMetricName,
     CustomMetricDescription,
     CustomSPName,
     ValueUnits,
     ParameterVersion)
VALUES
    ('Calls Resolved ON Contract',
     'Calculates percentage calls resolved ON contract.',
     'BPI_Contract_Resolved_Calls',
     'Percent',
     2);
```

where:

- The new monitor type is called Calls Resolved ON Contract.
- The associated stored procedure that writes out the monitor values is called BPI_Contract_Resolved_Calls.
- The display text for the measurement units for this monitor type is Percent.

Example - A Data Property

Suppose the monitor you wish to record is contained in a data property but this data property is not set as the Process Value of the process. You can set up a custom monitor that simply pulls out the data property and writes that as the monitor value instead of the Process Value.

Suppose your data definition contains the property `Discount`, and that is the property you wish to measure as your monitor.

Your stored procedure needs to locate the value for this property and return its value. The stored procedure is passed a number of parameters, and these include:

- `Data Definition ID`

This is the unique ID for the process's related data table entry in the `Data_Objects` table.

- `Data Instance ID`

This is the unique ID of this process's related data instance entry within the related data table.

You can use the `Data Definition ID` to get the name of the related data table, and then use the `Data Instance ID` to look up the actual data row within this related data table.

The Stored Procedure

Let's look at both an MSSQL example, and an Oracle example, for how to write the stored procedure.

MSSQL Example

Here is an example stored procedure for MSSQL:

```
CREATE PROCEDURE BPI_Discount_Value
    @Monitor_ID NVARCHAR(36), @Process_ID NVARCHAR(36),
    @ProcessInstance_ID NVARCHAR(36), @ProcessInstIdentifier NVARCHAR(40),
    @ProcessInstStartTime DATETIME, @ProcessInstStartTimeLongMillis BIGINT,
    @ProcessValue FLOAT, @DataDefinition_ID NVARCHAR(36),
    @DataInstance_ID NVARCHAR(36), @EventType NVARCHAR(12),
    @EventTime DATETIME, @EventTimeLongMillis BIGINT, @Idx INTEGER
AS
    DECLARE @DiscountValue FLOAT
    DECLARE @DataTableName NVARCHAR(128)
    DECLARE @DataSqlStmnt NVARCHAR(256)
BEGIN

    -- (1) Use the DataDefinition_ID to get the name of the data table.

    select @DataTableName = InstanceTable
        from Data_Objects do
        where do.model_id = @DataDefinition_ID;

    -- (2) Now use DataInstance_ID to look up the actual data record
    --      and pull out the discount column value.

    CREATE TABLE Temp_BPI_CustomTable (tValue FLOAT)
    SET @DataSqlStmnt = 'insert into Temp_BPI_CustomTable(tValue) ' +
        ' select Discount from ' + @DataTableName +
        ' where id = '' + @DataInstance_ID + ''''
    EXECUTE (@DataSqlStmnt)
    SELECT @DiscountValue = tValue from Temp_BPI_CustomTable
    DROP TABLE Temp_BPI_CustomTable

    -- (3) Now write the monitor value.

    EXECUTE METRIC_SEND_EVENT_V2 @Monitor_ID, @Process_ID, @ProcessInstance_ID,
        @ProcessInstIdentifier, @ProcessInstStartTime,
        @ProcessInstStartTimeLongMillis, @ProcessValue,
        @DataDefinition_ID, @DataInstance_ID, @EventType,
        @EventTime, @EventTimeLongMillis, @DiscountValue, @Idx

END;
```

where:

- This defines a stored procedure called `BPI_Discount_Value`.
- Step (1) shows the SQL to determine the name of the related data table. This is the data table that holds the process instance's related data record.
- Step (2) shows how to use this related data table name to locate the data record and pull out the discount attribute value.

The SQL you logically want to use at this point is:

```
select @DiscountValue = Discount
      from @DataTableName dot
      where dot.id = @DataInstance_ID;
```

However, MSSQL does not let you issue a select statement in this form where the table name is a variable. Hence you can create a temporary table and use the `Execute()` command to execute the select statement.

- Step (3) sends the monitor result to the Monitor Engine.



The above MSSQL example may have performance issues. The creating and deleting of temporary tables within MSSQL can have a significant impact on your system's performance!

Oracle Example

Here is an example stored procedure for Oracle:

```
CREATE or REPLACE PROCEDURE BPI_Discount_Value(
  Monitor_ID VARCHAR2, Process_ID VARCHAR2, ProcessInstance_ID VARCHAR2,
  ProcessInstIdentifier VARCHAR2, ProcessInstStartTime DATE,
  ProcessInstStartTimeLongMillis NUMBER, ProcessValue FLOAT,
  DataDefinition_ID VARCHAR2, DataInstance_ID VARCHAR2,
  EventType VARCHAR2, EventTime DATE, EventTimeLongMillis NUMBER,
  Idx NUMBER)
IS
  DiscountValue FLOAT;
  DataTableName VARCHAR2(128);
  DataSqlStmnt  VARCHAR2(256);
BEGIN
  -- (1) Use the DataDefinition_ID to get the name of the data table
  select InstanceTable into DataTableName
    from Data_Objects do
     where do.model_id = BPI_Discount_Value.DataDefinition_ID;

  -- (2) Now use DataInstance_ID to look up the actual data record
  --     and pull out the discount column value
  DataSqlStmnt := 'select Discount from ' || DataTableName || ' where id = :1';
  EXECUTE IMMEDIATE DataSqlStmnt INTO DiscountValue using DataInstance_ID;

  -- (3) Now write the monitor value.
  METRIC_SEND_EVENT_V2(Monitor_ID, Process_ID, ProcessInstance_ID,
    ProcessInstIdentifier, ProcessInstStartTime,
    ProcessInstStartTimeLongMillis, ProcessValue,
    DataDefinition_ID, DataInstance_ID, EventType,
    EventTime, EventTimeLongMillis, DiscountValue, Idx);
END;
```

where:

- This defines a stored procedure called `BPI_Discount_Value`.
- Step **(1)** shows the SQL to determine the name of the related data table. This is the data table that holds the process instance's related data record.

- Step (2) shows how to use this related data table name to locate the data record and pull out the discount attribute value.

The SQL you logically want to use at this point is:

```
select Discount into DiscountValue
   from :DataTableName dot
  where dot.id = :DataInstance_ID;
```

However, Oracle does not let you issue a select statement in this form where the table name is a variable. Hence you need to build the SQL statement in a variable, and then execute this variable passing in the DataInstance_ID.

- Step (3) sends the monitor result to the Monitor Engine.

The Custom Monitor Type Definition

Once the stored procedure has been installed into your database you need to tell the Monitor Definer about it.

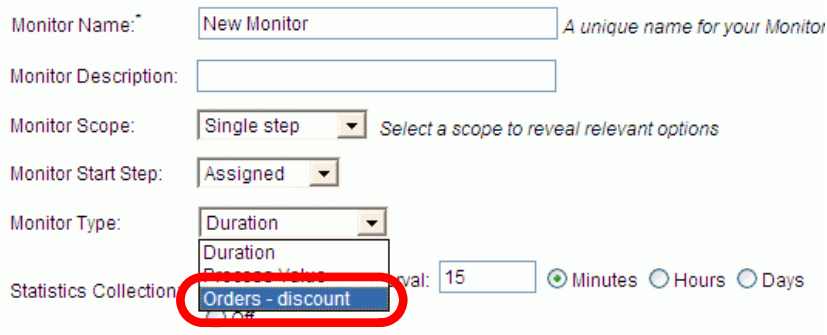
You create an entry in the `METRIC_CustomTypes` table defining a name for your custom monitor and linking this to your stored procedure.

For example:

```
INSERT INTO METRIC_CustomTypes (CustomMetricName,  
                                CustomMetricDescription,  
                                CustomSPName,  
                                ValueUnits,  
                                ParameterVersion)  
VALUES('Orders - discount',  
      'Calculates the percentage discount given on this order.',  
      'BPI_Discount_Value',  
      'Percent',  
      2);
```

When you defines a monitor in the Monitor Definer you are now able to select your stored procedure by selecting the Monitor Type of `Orders - discount`, as shown in [Figure 19](#).

Figure 19 Custom Monitor Type



The screenshot shows the 'Monitor Definer' interface with the following fields and values:

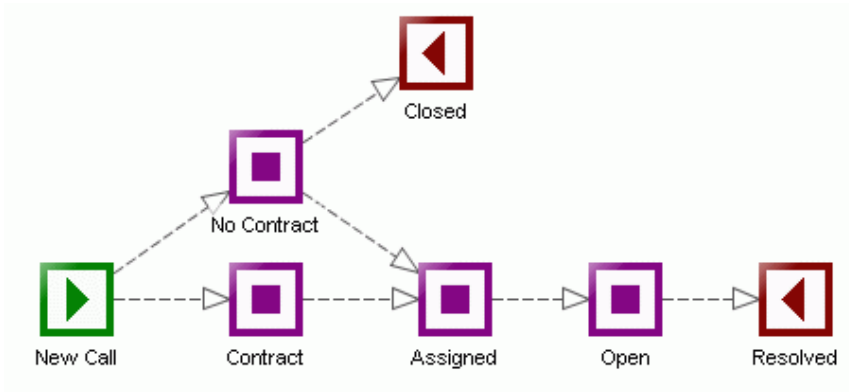
- Monitor Name: A unique name for your Monitor
- Monitor Description:
- Monitor Scope: Select a scope to reveal relevant options
- Monitor Start Step:
- Monitor Type: (dropdown menu open showing options: Duration, Process Value, **Orders - discount**)
- Statistics Collection Interval: Minutes Hours Days

The 'Orders - discount' option in the 'Monitor Type' dropdown is highlighted with a red circle.

Example - Percentage Path Process

Suppose you have a process that looks as shown in [Figure 20](#).

Figure 20 Process with Multiple Paths



This process is the Call System process from an earlier lab ([Lab - Defining Monitors](#) on page 41). It monitors support calls as they come into the call center. The calls include some that are under a support agreement and some that are not. What's more, the supervisor may choose to handle a non-contract call because he/she feels the caller is a potential customer and therefore worth looking after.

Your job is to measure the calls that are being processed and report the following percentages:

- The percentage of contract calls being resolved.
- The percentage of non-contract calls being resolved.
- The percentage of non-contract calls being closed.

In other words, you need to measure the different outcomes of the process and show these outcomes as percentages.

This requires defining some custom monitors.

Monitor Scope

You want to be able to measure the state of each process instance when it has completed. So you want to define some stored procedures that are run at the completion of each process instance. This means that when you come to configure these monitors within the Monitor Definer (once all the stored procedures are defined and in place) you set the monitor scope to be `Whole Process`. This ensures that the monitor is collected at the start and the end of each process instance.

The Stored Procedure(s)

You need to write a stored procedure for each outcome of the process. That is, you write a stored procedure that measures the process instances that end at the `Closed` step. You write another stored procedure that measures the process instances that terminate at the `Resolved` step having gone through the `Contract` step, and another procedure that measures the process instances that terminate at `Resolved` step having gone through the `NoContract` step.

But what is it that you measure? What value do you return as the monitor value? And how do you calculate this value?

You write the stored procedures to return the value of either `0` or `100`.

Let's consider the stored procedure that is going to measure the process instances that have terminated at the `Resolved` step having come through the `Contract` step. The stored procedure uses the process instance `Id` passed in, to find out whether this process instance actually terminated at the `Resolved` step. It does this by accessing the `Node_Instance` table and looks to see if there is a completed step instance for the `Resolved` step for this process instance. If the process instance did terminate at the `Resolved` step, then the stored procedure looks to see if the `Contract` step is also completed for this process instance. If both of these tests are true, then this process instance terminated at the `Resolved` step having also completed the `Contract` step, and therefore the stored procedure returns a value of `100`. If the process instance did not terminate at the `Resolved` step or did not complete the `Contract` step, the stored procedure returns a value of `0`.

By returning a monitor value of either 0 or 100, the Monitor Engine can collect all the monitor values. Every collection interval the Monitor Engine calculates, among other things, the average of these values. Thus, the Monitor Engine produces the average of your monitor values, which is indeed the percentage.

Contract Resolved Calls

The stored procedure for counting the instances of contract calls that are resolved looks as follows (for MSSQL):

```
CREATE PROCEDURE BPI_Contract_Resolved_Calls
    @Monitor_ID NVARCHAR(36), @Process_ID NVARCHAR(36),
    @ProcessInstance_ID NVARCHAR(36), @ProcessInstIdentifier NVARCHAR(40),
    @ProcessInstStartTime DATETIME, @ProcessInstStartTimeLongMillis BIGINT,
    @ProcessValue FLOAT, @DataDefinition_ID NVARCHAR(36),
    @DataInstance_ID NVARCHAR(36), @EventType NVARCHAR(12),
    @EventTime DATETIME, @EventTimeLongMillis BIGINT, @Idx INTEGER
AS
    DECLARE @RetVal FLOAT
    DECLARE @Count1 FLOAT
    DECLARE @Count2 FLOAT
BEGIN
    -- For this flowinstanceID, is the 'Resolved' step completed
    --                               and the 'Contract' step completed
    -- If so = set value to 100
    -- If not = set value to 0

    -- (1) If this is the start of the monitor, then simply return.
    --     NOTE: This means that no backlog information is available.
    IF (@EventType = 'Started')
    BEGIN
        RETURN;
    END

    -- Assume a false outcome
    SET @RetVal = 0

    -- (2) Did the process instance terminate at the 'Resolved' step
    --     (For this flowInstanceID, is the 'Resolved' step 'Completed')
    SET @Count1 = 0
    SELECT @count1 = count(ni.flowinstance_id)
        from node_instance ni, nodes
        where ni.node_id = nodes.node_id
        and ni.flowinstance_id = @ProcessInstance_ID
        and (nodes.nodename = 'Resolved' and ni.status = 'Completed')
```

```

-- Only do this check if this instance terminated at the 'Resolved' step.
IF (@Count1 != 0)
BEGIN
  -- (3) Check whether the 'Contract' step is also 'Completed'
  SET @Count2 = 0
  SELECT @Count2 = count(ni.flowinstance_id)
    from node_instance ni, nodes
    where ni.node_id = nodes.node_id
    and ni.flowinstance_id = @ProcessInstance_ID
    and (nodes.nodename = 'Contract' and ni.status = 'Completed')
END

-- (4) If it did go through the steps, set the return value to 100
IF (@Count1 != 0 AND @Count2 != 0)
BEGIN
  SET @RetVal = 100
END

-- (5) Now write the result to the monitor engine

EXECUTE METRIC_SEND_EVENT_V2 @Monitor_ID, @Process_ID, @ProcessInstance_ID,
  @ProcessInstIdentifier, @ProcessInstStartTime,
  @ProcessInstStartTimeLongMillis, @ProcessValue,
  @DataDefinition_ID, @DataInstance_ID, @EventType,
  @EventTime, @EventTimeLongMillis, @RetVal, @Idx
END;

```

where:

- Step (1) tests if this is the start of the monitor, simply return. You only wish to measure completed instances. This means that the Business Process Dashboard is unable to show any backlog numbers for this monitor.
- Step (2) tests that this process instance has actually completed at the Resolved step.
- Step (3) tests that, if the process instance has completed at the Resolved step, did it also complete the Contract step.
- Step (4) tests whether all conditions have been met. If so, it sets the return value to be 100.
- Step (5) sends the monitor result to the Monitor Engine.

Non-Contract Resolved Calls

The stored procedure for counting the instances of non-contract calls that are resolved looks as follows (for MSSQL):

```
CREATE PROCEDURE BPI_Off_Contract_Resolved_Calls
    @Monitor_ID NVARCHAR(36), @Process_ID NVARCHAR(36),
    @ProcessInstance_ID NVARCHAR(36), @ProcessInstIdentifier NVARCHAR(40),
    @ProcessInstStartTime DATETIME, @ProcessInstStartTimeLongMillis BIGINT,
    @ProcessValue FLOAT, @DataDefinition_ID NVARCHAR(36),
    @DataInstance_ID NVARCHAR(36), @EventType NVARCHAR(12),
    @EventTime DATETIME, @EventTimeLongMillis BIGINT, @Idx INTEGER
AS
    DECLARE @RetVal FLOAT
    DECLARE @Count1 FLOAT
    DECLARE @Count2 FLOAT
BEGIN
    -- (1) If this is the start of the monitor, then simply return.
    -- NOTE: This means that no backlog information is available.
    IF (@EventType = 'Started')
    BEGIN
        RETURN;
    END

    -- Assume a false outcome
    SET @RetVal = 0

    -- (2) Did the process instance terminate at the 'Resolved' step
    -- (For this flowInstanceID, is the end step 'Completed')
    SET @Count1 = 0
    SELECT @count1 = count(ni.flowinstance_id)
        from node_instance ni, nodes
        where ni.node_id = nodes.node_id
        and ni.flowinstance_id = @ProcessInstance_ID
        and (nodes.nodename = 'Resolved' and ni.status = 'Completed')

    -- Only do this check if this instance terminated at the 'Resolved' step.
    IF (@Count1 != 0)
    BEGIN
        -- (3) Check whether the 'No Contract' step is also 'Completed'
        SET @Count2 = 0
        SELECT @Count2 = count(ni.flowinstance_id)
            from node_instance ni, nodes
            where ni.node_id = nodes.node_id
            and ni.flowinstance_id = @ProcessInstance_ID
            and (nodes.nodename = 'No Contract' and ni.status = 'Completed')
    END
END
```

```

-- (4) If it did go through the steps, set the return value to 100
IF (@Count1 != 0 AND @Count2 != 0)
BEGIN
    SET @RetVal = 100
END

-- (5)

EXECUTE METRIC_SEND_EVENT_V2 @Monitor_ID, @Process_ID, @ProcessInstance_ID,
    @ProcessInstIdentifier, @ProcessInstStartTime,
    @ProcessInstStartTimeLongMillis, @ProcessValue,
    @DataDefinition_ID, @DataInstance_ID, @EventType,
    @EventTime, @EventTimeLongMillis, @RetVal, @Idx

END;

```

where:

- Step (1) tests if this is the start of the monitor, simply return. You only wish to measure completed instances. This means that the Business Process Dashboard is unable to show any backlog numbers for this monitor.
- Step (2) tests that this process instance has actually completed at the Resolved step.
- Step (3) tests that, if the process instance has completed at the Resolved step, did it also complete the No Contract step.
- Step (4) tests whether all conditions have been met. If so, it sets the return value to be 100.
- Step (5) sends the monitor result to the Monitor Engine.

Non-Contract Closed Calls

The stored procedure for counting the instances of non-contract calls that are simply closed (ignored), looks as follows (for MSSQL):

```
CREATE PROCEDURE BPI_Off_Contract_Closed_Calls
    @Monitor_ID NVARCHAR(36), @Process_ID NVARCHAR(36),
    @ProcessInstance_ID NVARCHAR(36), @ProcessInstIdentifier NVARCHAR(40),
    @ProcessInstStartTime DATETIME, @ProcessInstStartTimeLongMillis BIGINT,
    @ProcessValue FLOAT, @DataDefinition_ID NVARCHAR(36),
    @DataInstance_ID NVARCHAR(36), @EventType NVARCHAR(12),
    @EventTime DATETIME, @EventTimeLongMillis BIGINT, @Idx INTEGER
AS
    DECLARE @RetVal FLOAT
    DECLARE @Count1 FLOAT
BEGIN
    -- (1) If this is the start of the monitor, then simply return.
    --     NOTE: This means that no backlog information is available.
    IF (@EventType = 'Started')
    BEGIN
        RETURN;
    END

    -- Assume a false outcome
    SET @RetVal = 0

    -- (2) Did the process instance terminate at the 'Closed' step
    --     (For this flowInstanceID, is the end step 'Completed')
    SET @Count1 = 0
    SELECT @count1 = count(ni.flowinstance_id)
        from node_instance ni, nodes
        where ni.node_id = nodes.node_id
        and ni.flowinstance_id = @ProcessInstance_ID
        and (nodes.nodename = 'Closed' and ni.status = 'Completed')

    -- (3) If it did go through the 'Closed' step, set the return value to 100
    IF (@Count1 != 0)
    BEGIN
        SET @RetVal = 100
    END

    -- (4)

    EXECUTE METRIC_SEND_EVENT_V2 @Monitor_ID, @Process_ID, @ProcessInstance_ID,
        @ProcessInstIdentifier, @ProcessInstStartTime,
        @ProcessInstStartTimeLongMillis, @ProcessValue,
        @DataDefinition_ID, @DataInstance_ID, @EventType,
        @EventTime, @EventTimeLongMillis, @RetVal, @Idx
END;
```

where:

- Step (1) tests if this is the start of the monitor, simply return. You only wish to measure completed instances. This means that the Business Process Dashboard is unable to show any backlog numbers for this monitor.
- Step (2) tests that this process instance has actually completed at the Closed step.
- Step (3) tests whether the condition has been met. If so, it sets the return value to be 100.
- Step (4) sends the monitor result to the Monitor Engine.

The Custom Monitor Type Definitions

Once the stored procedures are installed into your database you need to tell the Monitor Definer about them and give them names.

You create an entry in the `METRIC_CustomTypes` table defining a name for your custom monitor and linking this to your stored procedure.

For example, to load in the three stored procedures, you issue the following commands:

```
INSERT INTO METRIC_CustomTypes (CustomMetricName,
                                CustomMetricDescription,
                                CustomSPName,
                                ValueUnits,
                                ParameterVersion)
VALUES('Calls Resolved ON Contract',
       'Calculates the percentage calls resolved ON contract.',
       'BPI_Contract_Resolved_Calls',
       'Percent',
       2);
```

```

INSERT INTO METRIC_CustomTypes (CustomMetricName,
                                CustomMetricDescription,
                                CustomSPName,
                                ValueUnits,
                                ParameterVersion)
VALUES('Calls Resolved OFF Contract',
      'Calculates the percentage calls resolved OFF contract.',
      'BPI_Off_Contract_Resolved_Calls',
      'Percent',
      2);

INSERT INTO METRIC_CustomTypes (CustomMetricName,
                                CustomMetricDescription,
                                CustomSPName,
                                ValueUnits,
                                ParameterVersion)
VALUES('Calls Closed OFF Contract',
      'Calculates the percentage OFF contract calls closed.',
      'BPI_Off_Contract_Closed_Calls',
      'Percent',
      2);

```

When you define a monitor in the Monitor Definer, you are now able to select your stored procedures by selecting one of the defined Monitor Types as shown in [Figure 21](#)

Figure 21 Custom Percentage Monitor Types

The screenshot shows the 'Monitor Definer' interface with the following fields and options:

- Monitor Name:** A unique name for your Monitor
- Monitor Description:**
- Monitor Scope:** Select a scope to reveal relevant options
- Monitor Start Step:**
- Monitor Type:**
 - Duration
 - Process Value
- Statistics Collection:**
 - Minutes
 - Hours
 - Days

The dropdown menu for 'Monitor Type' is open, and the options 'Calls Closed OFF Contract', 'Calls Resolved OFF Contract', and 'Calls Resolved ON Contract' are highlighted by a red circle.

Defining The Monitors

As there are three possible outcomes for the `Call System` process, you define three monitors. Each monitor must produce results for every process instance going through the `Call System` process, as this enables the Monitor Engine to produce an average. So you define three monitors, each with a scope of `Whole Process`.

You can set the collection interval to be whatever you require. Remember that the collection interval determines the frequency with which the percentage is calculated and updated.

With each monitor definition, you select the `Monitor Type` from the three custom monitor types you defined.

For example, to configure the monitor that measures the percentage of contract calls that are resolved, you define the monitor as shown in [Figure 22](#).

Figure 22 Monitor Definition - Resolved Contract Calls

The screenshot shows a form for defining a monitor. The fields are as follows:

- Monitor Name:** A unique name for your Monitor
- Monitor Description:**
- Monitor Scope:** Select a scope to reveal relevant options
- Monitor Type:**
- Statistics Collection:** On Off. **Collection Interval:** Minutes Hours Days
- Collected Data:**
 - Backlog - count
 - Backlog - process value
 - Throughput - count per hour
 - Throughput - process value per hour
 - Average process value
 - Minimum process value
 - Maximum process value
 - Weighted average process value
 - Process instance value
- Apply Filter:**
- Group Results By:**

You would define two more monitors, to measure the non-Contract Resolved calls and the non-Contract Closed calls.

Defining Thresholds

To see your percentages within the HPBPI Business Process Dashboard as dials, you can define thresholds for each monitor.



If you have configured the HPBPI integration with Business Availability Center then you cannot configure these statistical thresholds from within the Monitor Definer. Instead you would need to configure the display of these thresholds within the Business Availability Center dashboard.

Let's assume that the integration with Business Availability Center is not configured...

You can define thresholds that measure the recent average as this is the percentage. This allows you to set percentage values as the ranges for warning, minor, major and critical, as shown in [Figure 23](#).

Figure 23 Threshold Definition - Resolved Contract Calls

The screenshot displays the configuration for a threshold within the HPBPI Monitor Definer. The fields are as follows:

- Process Name: Call System (Deployed @ 14-Mar-2008 15:18:38)
- Monitor Name: Calls Resolved ON Contract
- Monitor Type: Calls Resolved ON Contract
- Monitor Scope: Whole process
- Threshold Name: ON Contract Calls Resolved (with a note: *A unique name for your Threshold*)
- Threshold Description: (empty text box)
- Threshold Type: Absolute value (dropdown menu with note: *Select a type to reveal relevant options*)
- Threshold Measure: Recent (radio button selected) with Average value (dropdown menu)
- Test Condition: Less than or equal to (radio button selected) / Greater than or equal to (radio button unselected)
- Warning Violation: 80 (with a green checkmark icon)
- Minor Violation: 60 (with a yellow warning icon)
- Major Violation: 40 (with an orange warning icon)
- Critical Violation: 20 (with a red 'x' icon)
- Violation Message: (empty text box)

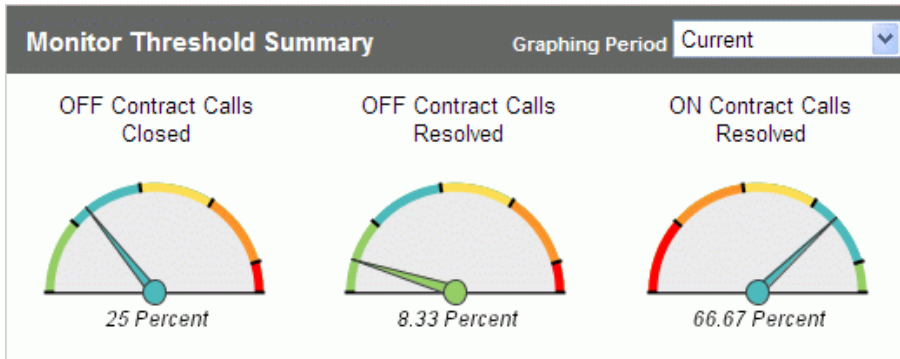
This threshold alarms whenever the percentage of resolved contract calls drops below 80 percent.

The HPBPI Business Process Dashboard

Now that the monitors and thresholds are defined, you can view the percentages within the HPBPI Business Process Dashboard. When you view the Call System process, the three threshold dials show the percentages of your calls.

For example, you might see the dials as shown in [Figure 24](#).

Figure 24 Business Process Dashboard - Percentages



These dials give you the current averages (or percentages in this case). You can use the `Graphing Period` pull-down to show the averages (percentages) over a longer period.

You can also click on any of these dials to see the average (percent) since the monitor was defined. This allows you to see the percentages over time and spot trends.

SQL Errors

When you write a stored procedure to produce a custom monitor value, your stored procedure is invoked when the Business Impact Engine updates the HPBPI database tables. The updating of the HPBPI tables and the running of your stored procedure all occur within a single database transaction. This means that if your stored procedure encounters a problem it may affect the Business Impact Engine transaction and thus the process instance. It all depends on the underlying database being used by HPBPI.

HPBPI on Oracle

If you are running HPBPI against an Oracle database then any problems within your monitor stored procedure are logged in the Business Impact Engine's log file, and this does **not** cause any problems the Business Impact Engine or the processing of the process instance.

The Oracle database system allows the stored procedure to throw an error, and have this treated separately within the overall Business Impact Engine's transaction. The Business Impact Engine is able to trap, and log, any errors and keep processing.

Obviously the values are not recorded correctly if the stored procedure is in error, but the process keeps running.

HPBPI on MSSQL

If you are running HPBPI against an MSSQL database then any problems within your monitor stored procedure cause the Business Impact Engine to abort the entire transaction. The error is logged to the Business Impact Engine's log file and it causes both the monitor and the current process instance to not be updated correctly.

Lab - Custom Monitors

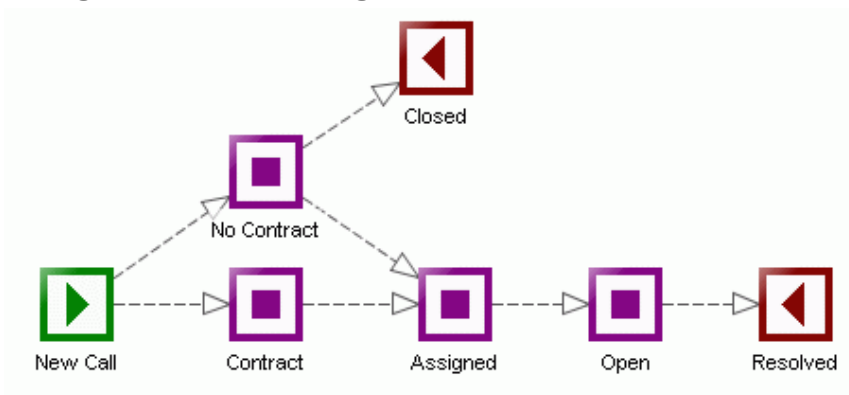
This lab gives you experience setting up custom monitors.

- ▶ This lab assumes that your HPBPI installation does not have the integration to Business Availability Center enabled. This is so that you can create statistical thresholds and then view them from within the HPBPI Business Process Dashboard.

The Call System Process

For this lab you are going to continue working with the Call System process you set up during the previous lab ([Lab - Defining Monitors](#) on page 41). The process diagram is as follows:

Figure 25 Process Diagram



This process monitors support calls as they come into the call center. The calls include some that are under a support agreement and some that are not. The supervisor may choose to handle a non-contract call because he/she feels the caller is a potential customer and therefore worth looking after.

The Required Monitors

Your job is to measure the calls that are being processed, and report the following percentages:

- The percentage of contract calls being resolved

- The percentage of non-contract calls being resolved
- The percentage of non-contract calls being closed

In other words, you need to measure the different outcomes of the process and show these outcomes as percentages.

The way to set up these monitors is fully explained earlier in this chapter in the section: [Example - Percentage Path Process](#) on page 88. You are to refer to that section for guidance as needed.

Your job in this lab is to do the following:

- Define the three required stored procedures, and define monitor type entries such that the Monitor Definer knows about your stored procedures.

To help you get started, the SQL for defining the stored procedure `BPI_Contract_Resolved_Calls`, and for defining the monitor value type `Calls Resolved ON Contract`, is located in the file:

```
labs\custom_ResolvedContractCalls.sql
```

This file contains the SQL for working with a MSSQL database.

- Define monitors to use these monitor types.

Define three monitors. All of these monitor are to be set to a scope of `Whole Process`. These monitors should invoke your stored procedures. Set the collection interval to be five minutes.

- Define a threshold for each of your monitors, to measure the recent average and set violations between the values of 0 and 100.
- Use the Process Simulator to send through more calls and view the percentages in the HPBPI Business Process Dashboard.

Make sure when you use the Process Simulator you ensure the `Instance ID` is greater than any previous run of the Process Simulator. That is, make sure that the Process Simulator injects new instances with new unique IDs.

When you have the HPBPI Business Process Dashboard showing the percentages, take some time to look at the other monitors that you defined in the first lab.

If you kept the Process Simulator running since the first lab, you should be able to use the HPBPI Business Process Dashboard to view the details of your monitors over time.

For example:

- In the HPBPI Business Process Dashboard, on the main Business Process & Resource Summary page - the page that shows the process diagram and the threshold dials - click on the tab marked: Monitors
- Click on the monitor called: Call Processing Time
By default this shows you a graph displaying the Average/Minimum/Maximum values for each collection interval.
- Select the Data Source for the graph to be: Completed - Count
This shows you the number of calls that have been resolved within each collection interval.
- Now select Chart by group to be: Yes
You now see the number of calls that have been closed within each collection interval, but now showing the numbers for each call priority.
This shows you the completion statistics since the first lab when you set up the Call Processing Time monitor.

Feel free to explore all your monitors using the HPBPI Business Process Dashboard and the many options available to you.

Well done! You have reached the end of the lab.

4 Further Topics

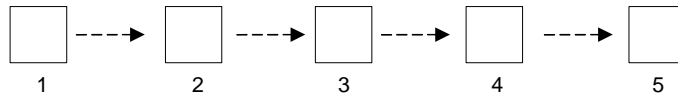
This chapter looks at additional topics to do with HPBPI monitors.

Monitor/Threshold Activation

When does a monitor definition become active? The answer is, the moment you click the `OK` button within the Monitor Definer. This means that monitors only begin to be recorded from the time you click the `OK` button when defining a new monitor.

Suppose you have a process as shown in [Figure 26](#).

Figure 26 Simple Process



and you already have some process instances that are sitting in step 3.

You then define a monitor to measure from the start of step 2 to the end of step 4.

The process instances that are currently in step 3 are past the start of the monitor so you do not see any start monitors for these process instances. When these process instances move out of step 4, end monitor records are generated.

When you modify a monitor definition and click the `Replace` button, all previous monitor history and violations are removed, and the monitor starts recording data afresh as if it were a brand new monitor definition.

Pre-Dated Events (BPI_GeneratedDate)

As just explained, a monitor definition becomes active from the moment you click the `OK` button within the Monitor Definer. If you then sent in business events containing `BPI_GeneratedDate` values that are earlier than when you defined the monitor, any process activity resulting from these events would **not** create monitor instances. This is because these events, and therefore the resultant process/step activity, would be seen as having occurred **before** the monitor had been defined.

Detecting Thresholds

When you modify a time-based threshold be careful if you update the violation time.

Suppose you define a monitor for the process shown in [Figure 26](#) on page 106, and this monitor measures the time it takes for instances to move from the start of step 2 to the end of step 4. You then define a threshold to raise a violation if any specific instance takes longer than 10 minutes.

To test this out, you then send in a process instance and progress it to step 3. You leave the process instance in step 3 for 10 minutes and wait for the violation to be issued.

After waiting for four minutes you think “Why am I waiting all this time?” so you decide to modify the threshold definition and update it to issue a violation if any specific instance takes longer than just one minute. You assume that you now only have to wait one more minute to see the violation appear.

You wait...and wait...and after 15 minutes of waiting, and seeing no violation, you think something has gone wrong.

You then assume that the violation is not going to happen, so you decide to progress the process instance and see what happens when it reaches the end of step 4. Sure enough, when the process instance exits step 4, you suddenly see a violation raised within the HPBPI Business Process Dashboard. This violation shows that the monitor instance did indeed take far longer than one minute to be completed.

So what happened?

It all has to do with the way the Monitor Engine looks for threshold violations while monitors are still active.

The Monitor Engine constantly monitors active monitor instances in case they exceed a threshold while still active. That is, if you want a violation raised when an instance takes longer than (for example) one hour to be completed, you don't want to be told this once the instance has completed. You want the violation raised the moment the instance has been running for more than your specified time period.

To identify active monitor instances that have been running for longer than a specified time period, the Monitor Engine monitors active monitor instances. But the Monitor Engine only needs to monitor the monitor instances that have started within the specified time period. In this example, the original threshold time period was set to 10 minutes. So every poll period the Monitor

Engine checks all monitor instances that have started within the last 10 minutes and looks to see if they are still running. If they are, then they have exceeded the threshold and a violation is raised.

The problem came about when you updated the threshold definition and changed the violation time period from 10 minutes to one minute. This meant that the Monitor Engine started monitoring active instances, but only within the last one minute period. Thus the original instance that you had started was out of the time range for active monitoring. This is why no violation was raised the moment the monitor instance took longer than one minute.

Of course, when the monitor instance finally does complete, the Monitor Engine detects this and raises a violation to say that the monitor instance took too long.

So, be careful about updating threshold times and making them shorter. This may cause violations to not be raised until the completion of some of the monitor instances.

Instance Violations

If you create a threshold and define threshold violations for Warning, Minor, Major and Critical, do not be surprised if you see a monitor instance raise (for example) a Warning violation followed by a Critical violation. That is, just because you have specified violations at all levels, an individual monitor instance may not raise them all. Let's explain...

Thresholds are checked every 60 seconds. (This time period is configurable within the HPBPI Administration Console.) When the threshold values are checked for a monitor instance, a violation is raised only for the highest (most severe) violation that has occurred within that 60 seconds. So, for example, if a monitor instance has raised a Minor violation, a Major violation and a Critical violation all within the last 60 seconds, the Monitor Engine only raises a single violation showing that the monitor instance has now gone critical.

Deadline Monitor Value is Fixed

With a deadline monitor, the value of the deadline property is read at the start of the monitor instance.

Suppose you have created a monitor and specified a `Deadline Property`. As each instance of this monitor is instantiated, the value within the deadline property is read from the process's related data definition, and stored with the monitor instance. Thresholds are then measured against this deadline attribute value held within the monitor instance. If the value within the deadline attribute (within the data definition) changes during the life of the monitor instance, the monitor instance does not see this. The monitor instance uses the attribute value it stored when it started.

Redeploying Processes/Monitors

Suppose you have defined monitors and thresholds on a deployed process. What happens if you need to redeploy the process?

If you redeploy a process that has monitors and thresholds defined, the new process is deployed and the deployer defines all the monitors and thresholds for this new process. In other words, the newly deployed process ends up having the same set of monitors and thresholds defined for it. However, the newly defined monitors and thresholds are new, and therefore the new monitors and thresholds do not inherit any history from the previous version of the monitors or thresholds.

When redeploying a process that currently has monitors and thresholds defined for it, the deployer may not be able to set up all the monitors and thresholds. For example, if you are deploying a new version of the process where you have removed a step, then the deployer is not able to redefine any monitors that were based on that step. So when you redeploy a process, the deployer tries to set up all the monitors and thresholds as defined on the superseded process definition. However, depending on what changes are within the newly deployed process, you may not end up with all the monitors in place. You need to check your monitor and threshold definitions within the Monitor Definer to ensure that they are set up as you require on the newly redeployed process.

If you undeploy a process definition then any subsequent redeployment results in the process being deployed but with no monitor definitions.



Once you have defined a set of monitors and thresholds for a deployed process, it is a good idea to export these monitor and threshold definitions from within the Monitor Definer (see [Exporting Monitors](#) on page 38) and save them to a file. This allows you to reinstate these monitor and threshold definitions should you ever have the scenario where you redeploy a process and find that the monitor and threshold definitions have not been applied to the newly deployed process.

Superseded Processes/Monitors

By redeploying process definitions that have monitors and thresholds defined, you can potentially end up with a lot of monitor and threshold definitions.

For example:

Suppose you are developing a process and you have defined five monitors. You then deploy this process and start up a number of process instances, which in turn start up some monitor instances. If you then make some changes to this process, redeploy the process and then start up some new process/monitor instances, you now have ten monitors active; the five for the superseded process and the five for the newly deployed version of the process. Every time you redeploy the process where the previously deployed process had active instances, you are adding another five monitor definitions to your HPBPI system.

So you need to be careful about your process development once you start adding monitors and thresholds.

Once you have redeployed a process definition that has monitors and thresholds defined, you might consider using the HPBPI Intervention Client to delete the superseded version of the process and its monitor and threshold definitions.

Deleting Process Monitors/Thresholds

If you undeploy a process from the HPBPI Business Impact Engine, the process is undeployed and any monitor or threshold definitions for this process are deleted. However, if there are still superseded versions of this process definition, then any monitor and threshold definitions for these superseded processes are still active and generating monitor statistics.

If you are deleting a process and you want to ensure that all monitor and threshold data is removed for this process, you need to remove the most recently deployed version of the process, and all superseded versions of the process.

Avoiding Notification Storms

When you are configuring your thresholds, and any notification subscriptions, you need to give some thought to how many violations this might end up sending. That is, configuring your thresholds to send out potentially hundreds of email notifications may not be helpful. This can be a particular issue when you set up an instance threshold, as this can lead to violations being raised for many individual instances. Indeed, you can also encounter the problem where HPBPI may send out violations faster than the email SMTP server can handle. This can lead to violation emails being delayed by the SMTP server.

When a violation is generated it is written to the `Metric_Fact_Alerts` table within the HPBPI database. The Monitor Engine is polling this table, and every 300 seconds, the Monitor Engine sends the new violations to the HPBPI Notification Server. You do not have to set up subscriptions in the Notification Server and send all violations to someone's email box. It may be better to configure HPBPI to send a single violation notification to an email box and then have that person access the `Metric_Fact_Alerts` table within the HPBPI database to see all the individual violation details.

When configuring your violations you might want to consider the approach of configuring more than one threshold. You configure the threshold that you actually want to monitor, and this causes violation to be raised as and when they happen. These violations are all written into the HPBPI database. You then configure an "overall" threshold - ideally a statistical threshold - that alarms when (for example) a maximum, or average, is above/below a certain threshold. You then set up an email notification subscription to this "overall" threshold violation.

Refer to [Monitor Notifications](#) on page 64 to see how you can also limit the actual number of notifications that get raised for each polling interval.

No Collection Interval Defined

It is possible to define a monitor to have no collection interval. For example, if you are measuring the time an instance takes to get to a particular step, and raising a violation if this time is greater than four hours, then you do not need to set any collection interval.

The HPBPI Business Process Dashboard uses the monitor's collection interval to make certain calculations. For example, the `Business Health Scorecard` page displays the most severe violation for a process over the last collection interval for each defined monitor. But what period does it use for monitors that have no collection interval defined?

For monitors that have no collection interval specified, the HPBPI Business Process Dashboard uses a period of:

`2 * Threshold Polling Interval`

The `Threshold Polling Interval` is configurable within the HPBPI Administration Console, and defaults to 60 seconds.

Business Process Dashboard and Violations

Business Health Scorecard Page

When you run the HPBPI Business Process Dashboard, the Business Health Scorecard page displays each deployed process name and shows a Business Process Monitor Status column.

The idea is that as business monitor violations occur, the Business Health Scorecard page reflects these in the Business Process Monitor Status column. Because instance violations (see [Instance and Statistical Thresholds/Violations](#) on page 36) have no way of being reset, the home page of the HPBPI Business Process Dashboard has been written to only include violations that have occurred within the last collection interval across all the thresholds defined for this process.

For thresholds where there is no collection interval configured see the discussion in [No Collection Interval Defined](#) on page 113.

So, the Business Process Monitor Status column represents the highest (most severe) violation that has occurred within the last collection interval across all the thresholds defined for this process.

Upgrading Custom Monitors

Prior to HPBPI version 2.20, the Monitor Engine used to store all dates and times in “date” attributes within the HPBPI database. This meant that for Oracle databases, these date/times were rounded to the nearest second. With the release of HPBPI 2.20 the Monitor Engine was enhanced to handle all dates and times in milliseconds.

In HPBPI version 2.20 all date/time attributes are still stored as they were prior to version 2.20. Additional attributes have been added to the Monitor Engine data tables to store the millisecond versions of each date/time attribute. For example, the `Metric_Fact_Values` table still has the attribute `TimeCompleted`. This table now also has the attribute `TimeCompletedLongMillis`, which holds the millisecond value (the number of milliseconds since January 1st 1970) corresponding to the date/time held in the `TimeCompleted` attribute.

Adding these additional millisecond attributes is something that affects the inner workings of the Monitor Engine and therefore you might wonder why you need to know about it. These changes have brought about a need to extend the custom monitor call interface.

With HPBPI version 2.20, the custom monitor interface has been extended to include these additional millisecond attributes, so there are now two versions of the custom monitor interface:

- The original interface.
- Version 2 of the interface.

The Original Interface

The original custom monitor call interface consists of the following three parts:

1. The stored procedure invocation.

The original version of the custom monitor stored procedure call has the following 11 parameters:

```
Monitor Id
Process Id
Process Instance Id
Process Instance Identifier
Process Instance start time
Process Value
Data Definition ID
Data Instance ID
Event type
Event time
Index
```

2. The send monitor call.

The stored procedure you invoke to send the monitor to the Monitor Engine is called `METRIC_SEND_EVENT` and this takes the following 12 parameters:

```
Monitor Id
Process Id
Process Instance Id
Process Instance Identifier
Process Instance start time
Process Value
Data Definition ID
Data Instance ID
Event type
Event time
Monitor Value
Index
```

3. The definition of the custom monitor.

The definition of your custom monitor is inserted into the `METRIC_CustomTypes` table. This is the entry that tells the HPBPI Monitor Engine which version of the custom monitor interface is being used by this custom monitor.

The key attribute is the `ParameterVersion` attribute. If this attribute is set to the value 1, or is not set, then the Monitor Engine invokes this custom monitor using the original version of the custom monitor interface.

Version 2 of the Interface

Version 2 of the custom monitor call interface consists of the following three parts:

1. The stored procedure invocation.

Version 2 of the custom monitor stored procedure call has the following 13 parameters:

```
Monitor Id
Process Id
Process Instance Id
Process Instance Identifier
Process Instance start time
Process Instance start time in milliseconds
Process Value
Data Definition ID
Data Instance ID
Event type
Event time
Event time in milliseconds
Index
```

The two parameters marked in “bold” are new in this version of the interface. They are not in the original version of the interface.

2. The send monitor call.

The stored procedure you invoke to send the monitor to the Monitor Engine is called `METRIC_SEND_EVENT_V2` and this takes the following 14 parameters:

```
Monitor Id
Process Id
Process Instance Id
Process Instance Identifier
Process Instance start time
Process Instance start time in milliseconds
Process Value
Data Definition ID
Data Instance ID
Event type
Event time
Event time in milliseconds
Monitor Value
Index
```

The two parameters marked in “bold” are new in this version of the interface. They are not in the original version of the interface.

3. The definition of the custom monitor.

The definition of your custom monitor is inserted into the `METRIC_CustomTypes` table. This is the entry that tells the HPBPI Monitor Engine which version of the custom monitor interface is being used by this custom monitor.

The key attribute is the `ParameterVersion` attribute. This attribute must be set to the value 2. The Monitor Engine then invokes this custom monitor using version 2 of the custom monitor interface.

Upgrading a Custom Monitor to use Version 2

A custom monitor defined using the original custom monitor interface will work on HPBPI version 2.20. You do not need to convert the monitor for it to continue working. However, if you would like your custom monitor to start recording its date/time details in milliseconds then you need to convert it to use version 2 of the custom monitor interface.

To upgrade an existing custom monitor that is using the original interface, to use version 2 of the interface, is very straightforward. You just need to change the following three things:

1. Change the stored procedure parameter list.
2. Replace the `METRIC_SEND_EVENT` call.
3. Replace the definition of the custom monitor.

Let's look at each of these and consider an example. The example shown is for an MSSQL custom monitor.

Change the Stored Procedure Parameter List

Your custom monitor stored procedures will be defined with the standard set of parameters, as follows:

```
CREATE PROCEDURE YouCustomMonitorSPName
    @Monitor_ID NVARCHAR(36), @Process_ID NVARCHAR(36),
    @ProcessInstance_ID NVARCHAR(36), @ProcessInstIdentifier NVARCHAR(40),
    @ProcessInstStartTime DATETIME, @ProcessValue FLOAT,
    @DataDefinition_ID NVARCHAR(36),
    @DataInstance_ID NVARCHAR(36), @EventType NVARCHAR(12),
    @EventTime DATETIME, @Idx INTEGER
```

You need to add the two new additional parameters `ProcessInstStartTimeLongMillis` and `EventTimeLongMillis` in the correct positions. The stored procedure call becomes as follows:

```
CREATE PROCEDURE YouCustomMonitorSPName
    @Monitor_ID NVARCHAR(36), @Process_ID NVARCHAR(36),
    @ProcessInstance_ID NVARCHAR(36), @ProcessInstIdentifier NVARCHAR(40),
    @ProcessInstStartTime DATETIME, @ProcessInstStartTimeLongMillis BIGINT,
    @ProcessValue FLOAT, @DataDefinition_ID NVARCHAR(36),
    @DataInstance_ID NVARCHAR(36), @EventType NVARCHAR(12),
    @EventTime DATETIME, @EventTimeLongMillis BIGINT,
    @Idx INTEGER
```

where the two new parameters are marked in “bold”.

Replace the METRIC_SEND_EVENT Call

Your stored procedure will be calling the `METRIC_SEND_EVENT` procedure to send the custom monitor value to the Monitor Engine. For example, your monitor send call might look as follows:

```
EXECUTE METRIC_SEND_EVENT @Monitor_ID, @Process_ID, @ProcessInstance_ID,  
    @ProcessInstIdentifier, @ProcessInstStartTime,  
    @ProcessValue, @DataDefinition_ID, @DataInstance_ID, @EventType,  
    @EventTime, @YourMonitorValue, @Idx
```

You need to replace this call with a call to `METRIC_SEND_EVENT_V2` and you need to add the two parameters `ProcessInstStartTimeLongMillis` and `EventTimeLongMillis` in the correct positions. Your monitor send call becomes as follows:

```
EXECUTE METRIC_SEND_EVENT_V2 @Monitor_ID, @Process_ID, @ProcessInstance_ID,  
    @ProcessInstIdentifier, @ProcessInstStartTime,  
    @ProcessInstStartTimeLongMillis, @ProcessValue,  
    @DataDefinition_ID, @DataInstance_ID, @EventType,  
    @EventTime, @EventTimeLongMillis, @YourMonitorValue, @Idx
```

where the changes are marked in “bold”.

You now need to update this custom monitor stored procedure within your database.

Replace the Definition of the Custom Monitor.

Your existing SQL call to define your custom monitor within the `METRIC_CustomTypes` table will look something like this:

```
INSERT INTO METRIC_CustomTypes  
    (CustomMetricName,  
    CustomMetricDescription,  
    CustomSPName,  
    ValueUnits)  
VALUES  
    ('Your Custom Monitor',  
    'Your Description',  
    'Your Custom Monitor SP Name',  
    'Your Units');
```

You now need to also specify the `ParameterVersion` attribute and set it to the value 2, as follows:


```
INSERT INTO METRIC_CustomTypes
  (CustomMetricName,
   CustomMetricDescription,
   CustomSPName,
   ValueUnits,
   ParameterVersion)
VALUES
  ('Your Custom Monitor',
   'Your Description',
   'You Custom Monitor SP Name',
   'Your Units',
   2);
```

where the changes are marked in “bold”.

You now need to apply this change to the `METRIC_CustomTypes` table in your HPBPI database.

Your custom monitor is now upgraded to Version 2 of the custom monitor interface.

