

HP

Business Process Insight

For the Windows® Operating System

Software Version: 7.50

Integration Training Guide - Modeling Processes

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 2004 - 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	Introduction	11
	Overall Architecture	12
	What Are You Trying to Achieve?	16
	How Data Events Progress a Process Instance	16
	Adding Service Status Feeds	22
	The Major Steps	24
	Monitoring The Business Impact Engine Statistics	25
	The Linkage To The Business Data - “Data Events”	26
	Suggested Methodology	28
	IT Service Dependencies	29
2	Process Definition	31
	Understand the Business Process	32
	Basic Analysis	32
	Business Process Execution Language (BPEL)	34
	Starting the HPBPI Modeler	35
	Logging On	35
	The Model Repository	37
	Logon Users	37
	Single User Repository	38
	Drawing The Process	39
	Creating A New Process	39
	Drawing the Steps Of The Process	40
	Lab - Drawing The Process	47
	Defining The Associated Data	49
	Creating a New Data Definition	49
	The Data Properties	50
	Deciding On The Data Properties	53

Lab - Defining The Associated Data	54
Relating The Data Definition To The Process	55
Lab - Relating The Data Definition	57
Configuring Progression Rules	58
The Rules Language	59
Style Of Progression	59
Switching Styles	62
No Implied Step Sequence	63
Lab - Progression Rules	64
Defining The Data Events	68
Define Each Event	68
Lab - The Data Events	72
Configuring The Event Subscriptions	73
Setting Up A Subscription	74
Lab - Event Subscriptions	83
Summary	89
Deploying the Process	90
The ToDo List	90
Viewing the Process	92
The Business Process Dashboard URL	92
Logging On	93
Testing the Process	94
Lab - Testing the Orders Process	95
Deploying the Process	95
Viewing the Deployed Process (HPBPI Dashboard)	95
Generating Data Events (Process Simulator)	96
Back in the HPBPI Business Process Dashboard	97
Injecting More Events	97
3 Using BAC	99
Configuring the Connection to BAC	100
General Settings	101
Data Samples Settings	102
Service Source Settings	103
A Worked Example	107

Configure the Integration.....	107
The CIs within BAC	109
Data Samples	111
Service Source	117
Summary	123
4 Advanced Process Definition.....	125
Actual Step Ordering.....	126
Trapping Steps Out Of Sequence.....	126
Steps With No Arcs.....	127
Active Processes With No Active Step(s).....	128
Activity Steps Starting Processes	129
Junction Steps	131
The Model Repository	133
Repository Startup	133
Importing And Exporting Definitions.....	134
Exporting.....	134
Importing.....	135
Repository Explorer	136
Running the Repository Explorer	136
View Details	137
Recycle Bin	137
Exporting Definitions	138
Deployment	140
Superseded Processes.....	140
Intervention Client	142
Multiple Unique IDs	143
Handling Out of Sequence Events	145
Lab - The Process Seems To Stop Working.....	148
The Process	148
Testing The Process	150
Altering The Process Definition.....	151
Testing The New Process	152
The Explanation	153
5 Advanced Progression Rules.....	155

The Language	156
The Grammar	156
Data Definition Level Methods	157
Data Property Level Methods	158
Comments Within Progression Rules	161
Single Start/Complete Condition	162
Steps Re-Starting/Completing Too Often	163
Complex Start Rules	165
Example 1	165
Example 2	167
Out of Sequence Business Data Events	168
Typical Progression Rules	168
Well Defined Start and Complete Conditions	171
Summary	175
6 Creating A Business Process	177
Initial Investigations	178
Initial Process Definition	178
Initial Data Definition	179
More Detail Required?	180
Stage Two Investigations	181
Lab - Defining the Basic Process Diagram	185
Data Requirements	186
Data Normalization	186
Data Used To Progress The Process	189
Additional Data Properties	189
Lab - Defining The Data/Progression Rules	192
Data Events	197
Lab - Defining Events/Subscriptions	197
Lab - Deploy/Test the Insurance Claim Process	200
Services	203
Lab - Defining IT Services Dependencies	204
7 Using Non-BAC Services	205
HPOM Integration	206
Architecture	206

Configuring The HPBPI HPOM Integration.....	208
The HPBPI Modeler	209
SOA Manager Integration.....	219
Architecture.....	219
Configuring the HPBPI SOA Manager Integration	221
The HPBPI Modeler	222
Standalone Services.....	231
Defining a New Service	231
Using a Standalone Service	232

1 Introduction

This chapter looks at the overall picture that you are trying to achieve by modeling your business process using HP Business Process Insight (HPBPI).

Overall Architecture

You can think of a running HPBPI system as having the following major parts:

- The Business Process

This is a high-level process model that represents a particular business process within your organization. This high-level process model is created using the HPBPI Modeler and is then deployed to the HPBPI Engine.

- The HPBPI Business Impact Engine

The Business Impact Engine runs your deployed process model. For the Business Impact Engine to maintain anything interesting about your process model (such as: the number of orders coming through your application systems, who the customers are that are placing orders, etc.) you need to be able to feed information about the data activities happening in your underlying applications. That is, you need some “data feeds” from your actual applications that are processing and running your business.

Remember, when you define a process model and deploy it into the HPBPI Business Impact Engine you are not using it to run your business. This process model is being used to monitor your business, and so, to be able to monitor your business the Business Impact Engine needs to have data feeds.

- The Data Feeds

The data feeds provide a stream of “events” to tell the Business Impact Engine what is happening in your real world.

For example, you might have a data feed to signal whenever someone places a new order over the Web site. You might have another data feed to signal when this order has been shipped. These events then enable the HPBPI Business Impact Engine to maintain instances of your process model for each order, and to show you where each order is within the defined process model.

These event feeds can come from databases, files, application systems, Web sites, JMS etc.. They need to be configured by someone who understands the underlying data sources and the HPBPI Event Handler. Configuration these event feeds is covered in detail in the *HPBPI Integration Training Guide - Business Events*.

- The IT Infrastructure Status Feeds

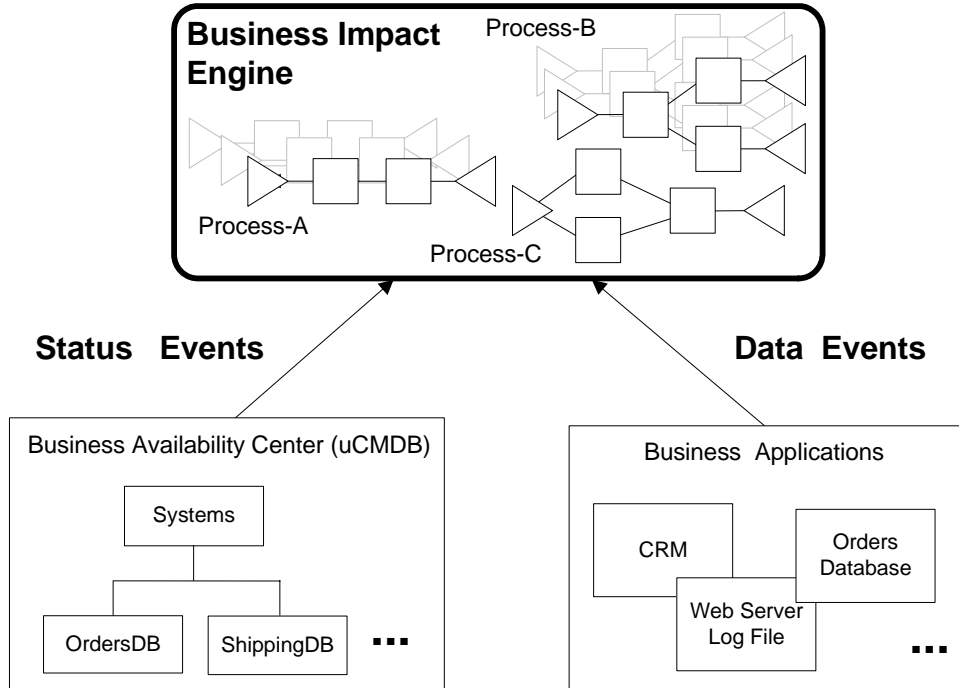
Once you have configured the HPBPI Business Impact Engine to monitor your business process and collect a whole load of information, that might be all you need. That might tell you what you need to know and help you see more clearly your business volumes, and where things are taking time within your business, etc.. But there is a further step you can take with HPBPI.

You can configure the HPBPI Business Impact Engine to take direct “status feeds” from HP Business Availability Center. Then, if there is an IT service failure, the HPBPI Business Impact Engine is able to show you how much business is affected by this failure, and thus the cost of this failure to your business.

As well as being able to receive status feeds from HP Business Availability Center, HPBPI can also be configured to receive status feeds directly from HP Operations Manager (HPOM) and/or Service Oriented Architecture (SOA) Manager.

A running HPBPI system might look something like this...

Figure 1 High-Level View of Components



where:

- A number of different processes have been modeled and deployed into the Business Impact Engine
- Status feeds are coming in from HP Business Availability Center advising the Business Impact Engine of any IT failures.

The status feeds are coming from the Universal CMDB (Configuration Management Data Base) within HP Business Availability Center.

- Data feeds are coming in from the underlying databases and application systems advising the Business Impact Engine of the data activity happening within these applications

- Notice that, as well as showing multiple processes, the diagram shows multiple instances of Process-A

If Process-A is a business process model that represents (for example) “Vendor Payments” then there is a separate instance of the process for each active vendor payment being handled. One process instance for each vendor payment.

At any one time the Business Impact Engine may be maintaining many parallel instances of any deployed process model.

Let’s now consider this in more detail...

What Are You Trying to Achieve?

Before jumping in and looking at the actual tools for creating process models and setting everything up, let's consider the above architecture in more detail. Specifically let's look at how things work when everything is in place and configured.

How Data Events Progress a Process Instance

Suppose you have set up the following:

- You have defined a process model that tracks orders through your business.
- For this process you have defined the data that you would like to maintain for each order.

For example:

— Order_ID

As assigned by the orders application system.

— Value

The actual dollar value of the order.

— Cust_ID

The ID of the customer who has placed this order.

— Credit_Status

Assigned true/false based on the customer's credit status.

— Ship_ID

The ID assigned by the shipping system when the order is shipped.

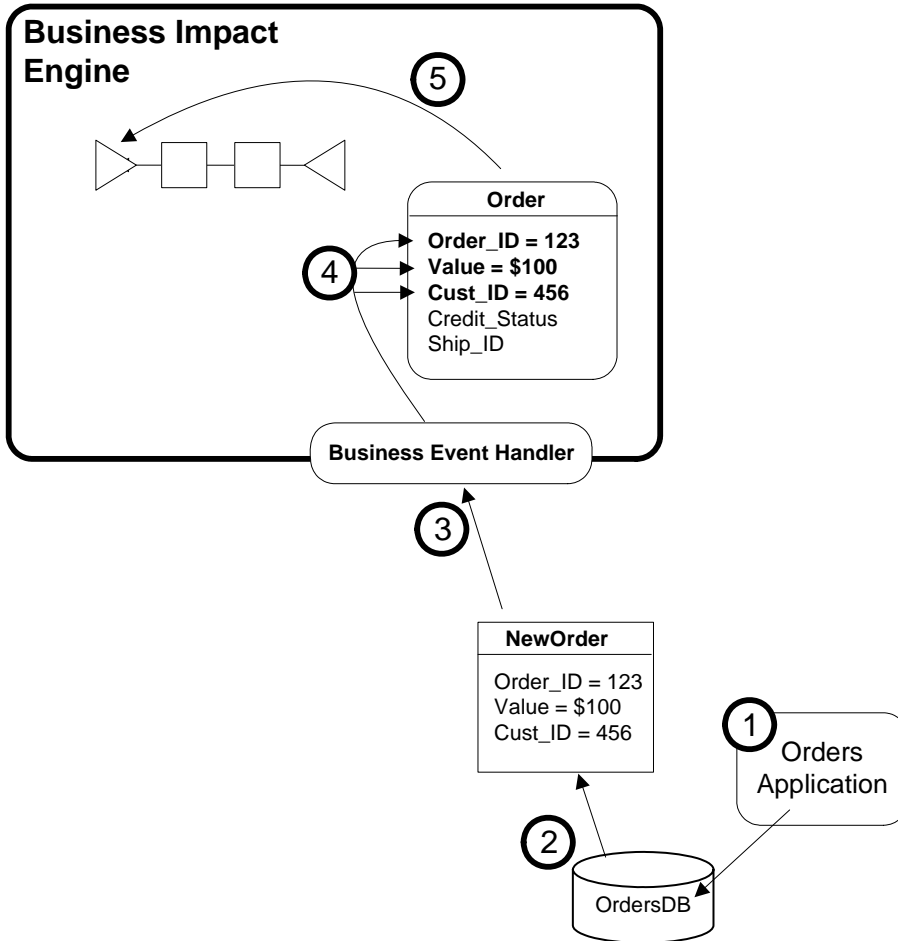
- You have then defined how the process knows where it is, based on values within this set of data attributes. (These are called **progression rules**.)

For example:

Once the `Credit_Status` field contains a value then you must have completed the "Check Credit" step in the process. If the `Credit_Status` value has changed to "false" then you must have entered the "Credit Check Failed" step within the process.

- You have then deployed this all into the HPBPI Business Impact Engine.
- Someone needs to have configured the underlying application systems to send in data events whenever new orders come in from the orders application.
- And the system runs something like this...

Figure 2 Application Data Feeding the Orders Process



where:

1. Someone rings up and places an order with your orders department.

The person taking the order, enters it into the orders system, using the orders application that they have always used. The system then assigns an order number - for example: order number 123.

The orders application then stores this order into its database.

2. Whenever your users accept a new order you want to trigger an event to be sent into the HPBPI Business Impact Engine to say that a new order has been placed.

You would like to send an event into the HPBPI Business Impact Engine specifying the order number (in this case 123), the dollar value for this order (for example: \$100), and the customer ID that has placed this order (for example: 456). You do not have to include every detail about the order, just the key details that are useful when producing reports. In this example you are mainly wanting to be able to report the number of orders and their value. By including the customer ID with each order this allows any reports to look up customer details (such as Name, Address, Phone...) direct from your existing customer database system.

Triggering an event for each new order is something that needs to be configured into the orders database. You could either configure an SQL trigger, or maybe the data table you need to monitor maintains a timestamp column. These sort of decisions require someone who understands the orders application and database structure.

3. You need to set up an adapter to monitor the data table and send the data into the HPBPI Business Impact Engine as a properly formed "event". Full details on how to configure such adapters are found in the *HPBPI Integration Training Guide - Business Events*.

So, the phone orders clerk accepts a new order, enters it into the orders system. Your adaptor notices the creation of this new order and send a data event into the HPBPI Business Impact Engine (passing through the HPBPI Business Event Handler component).

4. When designing a business process model, you lay out the steps that your process goes through. For example: your process might be describing the path that your orders take as they are processed within your company. You might accept a new order, carry out a credit check and then ship the product.

Clearly you want the HPBPI Business Impact Engine to monitor each order individually. Thus the HPBPI Business Impact Engine can instantiate a new process instance for each order that comes into the system. To do this, when you define a business process model you define a set of data attributes that you want the HPBPI Business Impact Engine to maintain for each instance of the process.

In the diagram you see that, for each instance of the process, you are maintaining the set of data attributes:

```
Order_ID  
Value  
Cust_ID  
Credit_Status  
Ship_ID
```

Thus when the Business Impact Engine receives the event saying that a new order has occurred, it instantiates a new instance of the process and stores with it the data from the event.

5. Once the data event has come into the Business Impact Engine, and data for that instance has been allocated and updated, the Business Impact Engine then uses that data to decide whereabouts in the process diagram this instance actually is.

When you defined the process model you also defined progression rules that determine where you are in the process diagram based on the values within the data for this process.

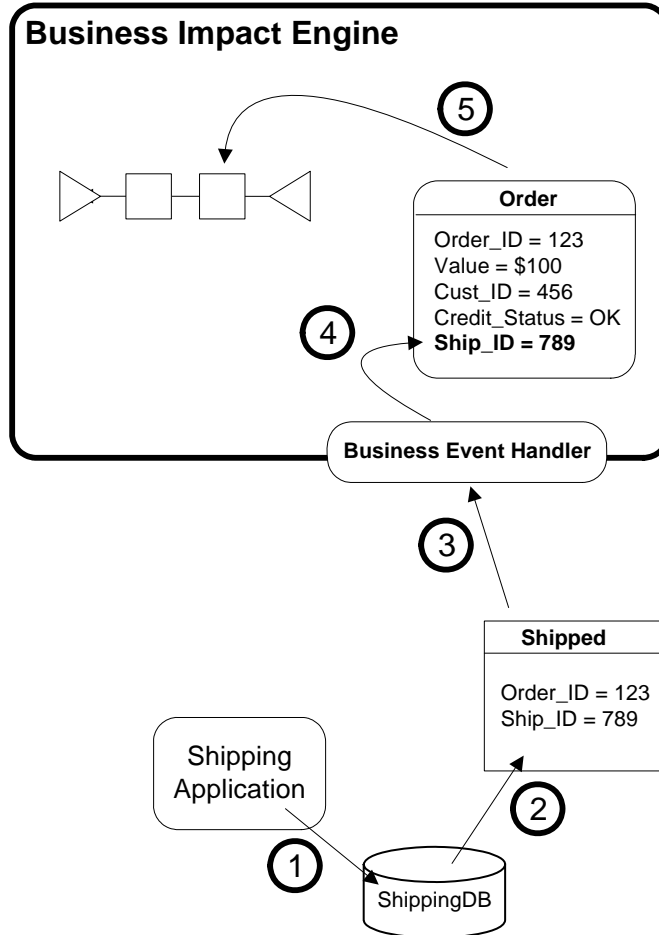
For example, in the above diagram the progression rules would simply say that when a new order comes in, the process instance is positioned at the first step. In other words, when a new order comes in, the first step within the process instance is considered “active”.

Hopefully you can see that with HPBPI you can design a process model and the HPBPI Business Impact Engine can then monitor all instances of that process. In this example, you are monitoring every order that is coming in - monitoring its value and the customer ID.

For this to work, someone had to configure the underlying orders application to be able to tell you when new orders have appeared.

Suppose you have also configured your shipping system to be able to tell you every time an order is shipped. When an order is shipped you would have the following scenario:

Figure 3 Shipping Updates Feeding the Orders Process



where:

1. The shipping department runs their usual application software to mark that an order has been shipped.
2. This writes to (probably) a table in a database.
3. You have an adaptor configured to monitor this data table and, as an order is shipped, the adaptor sends a data event containing the order ID and the corresponding shipping ID.

4. When the HPBPI Business Impact Engine receives this event it matches it to the existing order object for `Order_ID=123` and updates the `Ship_ID` attribute to the value from the incoming event (for example: 789).
5. The HPBPI Business Impact Engine then applies the progression rules defined for each step in the process to see if this change to the data attributes causes the process instance to move to a new step.

In this case, the assignment of the `Ship_ID` would cause the process to say that the step called “Ship Order” is now complete.

So you have the HPBPI Business Impact Engine monitoring the orders as they move through your business. This is excellent! You can now use the HPBPI Business Impact Engine statistics to generate reports to show information such as the volume of orders each day, the relative time that an order spends being shipped versus the time it takes to check the credit status of a customer...or whatever.

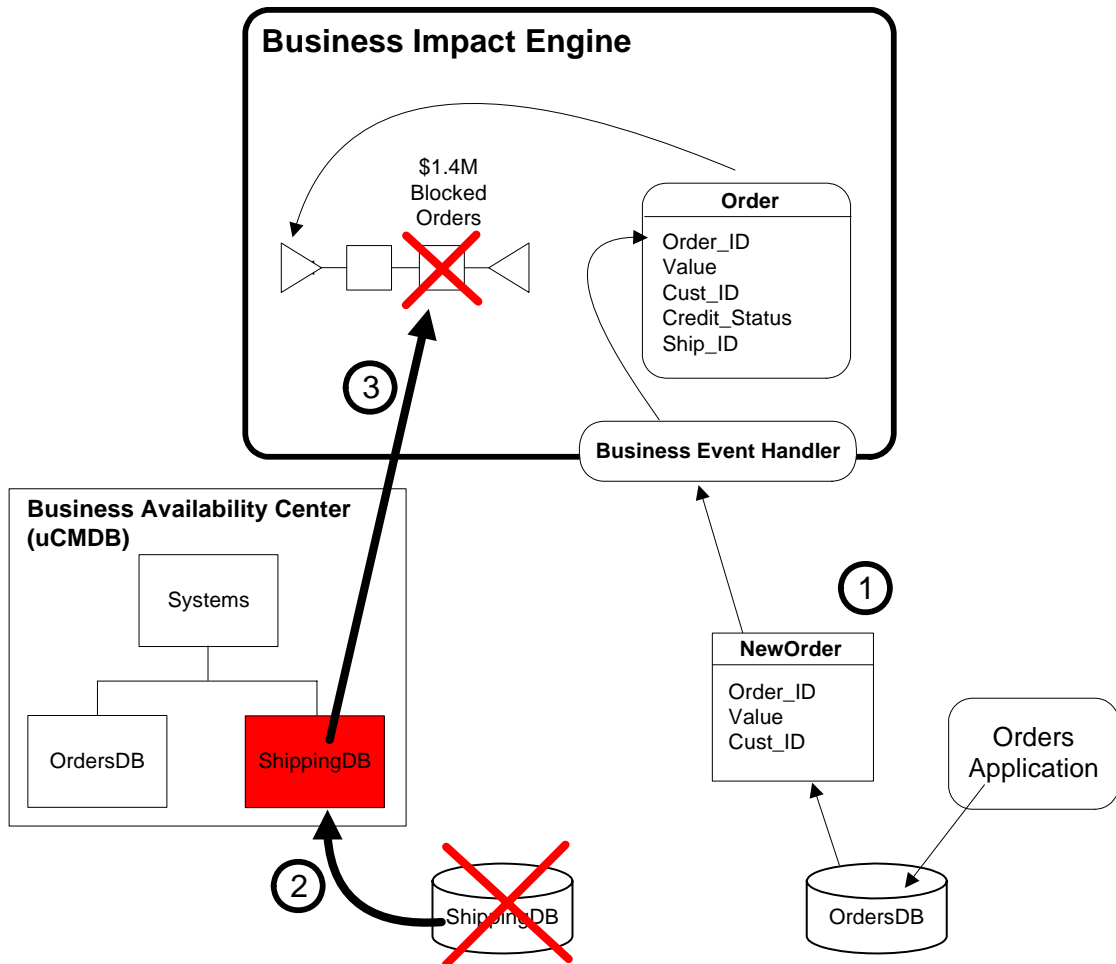
But you can also take this one step further...

Adding Service Status Feeds

The HPBPI Business Impact Engine is able to accept status feeds from your IT Infrastructure. So in your example, if the shipping database becomes unavailable you want the Business Impact Engine to tell you the volume of orders that are affected.

It looks something like this:

Figure 4 Receiving Service Status Feeds



where:

1. New orders are being entered into the system and monitored by the HPBPI Business Impact Engine.
2. The shipping database is being monitored by HP Business Availability Center. The shipping database suddenly becomes unavailable.
3. All process steps dependent upon the shipping database are flagged as being unavailable. You are then able to see the impact that this is having on your business.

For example, you can see the volume of orders waiting to be shipped, the list of affected customers, etc..

So, by adding the service status feeds you are not only able to have HPBPI monitor your business volumes and throughput, but you are able to report actual business impact when any underlying IT services become unavailable.

The Major Steps

Let's now start to generalize the example that you have just walked through and outline the main steps that are needed to monitor your business processes.

The major steps are:

- The business process model
 - Define the process diagram.
 - Define the associated data attributes to be maintained for each instance of this process.
 - Define the progression rules that allow the process model to determine where it is, based on the state of the associated data.
- The linkage to the business data - “data events”
 - Define the data events that are to be received from the underlying data applications.
 - Actually configure these events into the underlying data applications such that they are being generated correctly.
 - Configure adapters to monitor these events and send them into the HPBPI Business Impact Engine.
- The linkage to any underlying IT infrastructure - “status events”
 - Simply enable the integration with HP Business Availability Center. Your process models automatically receive these IT status events and are able to show any business impact.
- Deploy the process model to the HPBPI Business Impact Engine
- Monitor the Business Impact Engine statistics

Let's just talk a bit further about a couple of these points...

Monitoring The Business Impact Engine Statistics

The Business Impact Engine monitors your business processes. It maintains the data associated with each process instance. It is also able to show when a process step becomes impacted due to an IT service becoming unavailable. The question is: “How do you see all this?”

HPBPI Business Process Dashboard

The HPBPI Business Process Dashboard is a Web application that presents the Business Impact Engine statistics to the screen. The HPBPI Business Process Dashboard allows you to view each process that is deployed to the HPBPI Business Impact Engine, and to drill down to see each instance of a process. It also shows you the associated process instance data and impact information.

To run the HPBPI Business Process Dashboard, point your Web browser at the URL:

```
http://server:44080/hpbpi-bpd
```

where:

- *server* is the name of the server on which HPBPI is installed
- *44080* is the default port number on which the Tomcat servlet engine is running (on the HPBPI server)

The Linkage To The Business Data - “Data Events”

This is the step where you define the set of events that are to come in from the underlying data applications. These events feed into the Business Impact Engine and provide the data to update the process instance, and thus enable the process instance to determine where it is within the process diagram.

When you are creating the process model within the HPBPI Modeler you need to define the events that are to come into the Business Impact Engine at run time, and how you want them to affect your process. But someone actually needs to go and implement these events. That is, someone must physically set up some adapters to listen on the underlying application systems and create these data events.

So when it comes to defining the list of events that are going to come into your process there are two parts to it:

1. The process needs to know the list of incoming events

This includes things such as the event name and the attributes within the event - their name and type, etc.

2. Adapters need to be set up to produce these data events

Someone needs to go to the underlying application systems and determine where and how to trigger the necessary information, and then set up an adapter to generate this information as an HPBPI data event.

The obvious question is: “Which do you do first?”

Do you first configure into your underlying application systems every event that you think any process will ever need, and then simply refer to these within the process modeler? This the “bottom-up” approach. Or do you first define the list of events that you believe you need to drive your process and then go out to the application systems and get these events configured? This is the “top-down” approach.

The answer is that you can do it either way. However...

Bottom-Up Event Definition

The bottom-up approach probably appears to be the most logical. If you first define all the required events within the data applications, then it becomes quite easy to define the process as you know what events you are going to receive.

However, if you just go to your application people and say “Would you please define a whole load of events and have these generated every time something important happens in our business” they will probably not know what to do. They could no doubt invest the next year configuring every event imaginable...but would that be a good use of time? Probably not.

Top-Down Event Definition

It is probably better to consider the top-down approach. This means that the business process designer, whilst designing a specific process model, thinks carefully about the kind of data events that they would need to drive this process. They can then go to the applications people and say “Would you please see if you can generate these events from our data systems.” The application people now have a specific set of events on which to focus their attention.

It may be that the application people determine that they cannot implement every event exactly as specified - in which case the process designer needs to sit down with them and discuss alternatives and possibly make alterations to the process design. But this helps everybody focus on the minimum set of events required to drive the business process.

Suggested Methodology

Here is a suggested order in which to carry out the tasks of designing a business process model:

- 1. Design the business process model**

Use the HPBPI Modeler to draw out your high-level business process.

- 2. Define the data to be maintained for each process instance**

Define the data attributes that you wish to maintain for each process instance.

- 3. Define the progression rules**

Define how a process instance progresses as the data attributes change state during the life of the process instance.

- 4. Define the data events**

Define the events that you expect to receive from the underlying application systems. Define the name of each event and its data content.

You then need some consultation with the applications people to see if these events can be generated as you expect. For more details on how to generate these actual events refer to the HPBPI *Integration Training Guide - Business Events*.

- 5. Deploy the process and test it**

Use the HPBPI Modeler to deploy the process definition into the HPBPI Business Impact Engine. You can use tools such as the Process Simulator (contributed utility) and the HPBPI Business Process Dashboard (both described later in this manual) to test out the behavior of your process, before then connecting in the actual application system adapters to generate the real data events.

This is just a suggested order for defining processes. You might find that your preference is to interchange steps 3 and 4, and define the events before defining the progression rules. It actually doesn't matter. Each person can develop the order that best suits them. But you can use the above steps as your starting point.

Be aware that this process is iterative. You will revisit steps (perhaps multiple times) as you discover more about your process and as you determine more about the data that comes from the underlying application systems.

IT Service Dependencies

So what about the IT service dependencies? How do you configure them within HPBPI? Actually...you don't! Let's explain...

All your IT infrastructure is modeled within the uCMDB of Business Availability Center. When you enable the integration between HPBPI and Business Availability Center (from within the HPBPI Administration Console) HPBPI synchronizes your process, and its steps, with the Business Availability Center uCMDB. This means that configuration Items (CIs) are created within your uCMDB. You will have a CI for your process and this will contain dependent CIs for each of the steps of the process. You then configure, within the uCMDB, the IT dependencies for each of the steps within your process. As you deploy new processes within HPBPI, these automatically create process and step CIs within the uCMDB.

So all your IT dependencies are modeled within the uCMDB.

But what if you have a standalone HPBPI installation? What if you do not have an integration with Business Availability Center?

If you do not have an integration with Business Availability Center then you have the option to configure IT dependency links with HP Operations Manager and/or HP SOA Manager. Refer to [Chapter 7, Using Non-BAC Services](#) for details of how to set up IT dependencies for a standalone HPBPI installation.

2 Process Definition

This chapter looks at the HPBPI Modeler and the steps to defining a business process.

Understand the Business Process

The first thing that you need to do is gain an understanding of the actual business process you wish to monitor.

Basic Analysis

Business processes are typically very complex, but your aim is to identify the main phases that occur during the life cycle of the real business process.

Remember you are wanting to **monitor** the process, not **run** it.

So your aim is to try and produce a high-level summary of the underlying business process.

For example, you might be wanting to monitor the company's order fulfillment process. Now there are probably many many steps, and departments, involved in handling an order. But for monitoring purposes you may identify that the main phases for your orders are:

- The order comes in
- A credit check is carried out
- If OK, the order is picked in the warehouse
- The order is then shipped

This then allows you to monitor things such as:

- The volume of orders for each day
- The value of your orders
- Average time to credit check clients
- Percentage of failed credit checks
- Warehouse time to pick an order
- and so on

The idea is to understand the real business process but express it in simplified terms.

The good thing about defining a high-level business process within HPBPI is that it is not actually running your business, it is monitoring it. So if you create a process within HPBPI and you later find that it does not capture enough monitoring statistics to really be of interest...that's OK! You can simply extend the process definition and try that. Indeed, you might define a simple process, deploy it to HPBPI and when you see the reports you decide that your business is running just fine and you decide that it is not worth any more investment to monitor that particular business process, allowing you to focus your time monitoring other parts of your business. Or, you may find that the process does indicate some issues and you decide that it is worth investing time to extend the HPBPI process definition and gather more statistics about your business.

The key here is that your HPBPI process definition should start out simple - with only a few steps - and if it turns out that you need to gather more detailed statistics...you can! The process can always be extended if you so desire. You do not have to get the process right the first time!

As you are gaining an understanding of your real business process, it is a good time to be finding out about the underlying application systems. The person/people responsible for setting up the HPBPI business event adapters that are to generate the data events to drive the monitoring of this process, need to know the possible places where these events might come from. It's a good idea to make notes about the possible data sources/systems for each step within your HPBPI process definition.

Another important step is to keep asking yourself "What statistics do you want to get from HPBPI when this high-level process is deployed and running?" This helps you to decide what data you want the process to maintain for each instance. This also helps to define what data properties you need to get from the underlying application systems.

So, important steps:

- Gain an understanding of the actual business process and produce from this a high-level version of the business process to represent the main phases.
- Ask yourself the question:
"What statistics do I want to get from this process?"
- List out the possible sources for data events for each step within your process.

Business Process Execution Language (BPEL)

You may find that the customer has some, or all, of their business processes defined within a third party business process design tool. In which case they may be able to provide the process definition as a BPEL export file. The HPBPI Modeler is able to import BPEL processes and create a corresponding HPBPI process definition.

Importing the BPEL process can provide you with a good starting point when trying to create the high-level business process within HPBPI. However, the BPEL may well contain so many process steps that it might, at first, appear as an overwhelmingly large and detailed process diagram. You typically need to go through the resultant HPBPI process definition and refine it to produce your high-level business process.

Even if you do import the BPEL process, it only provides you with the process diagram. You still need to define the required data feeds and data definitions for this process to be able to monitor the underlying process. So you still need to ask yourself the questions outlined in the section [Basic Analysis](#) on page 32

For details about importing BPEL processes, refer to the *HPBPI Integration Training Guide - Importing BPEL*.

Starting the HPBPI Modeler

To run the HPBPI Modeler:

Start->Programs->HP->HP Business Process Insight->Modeler

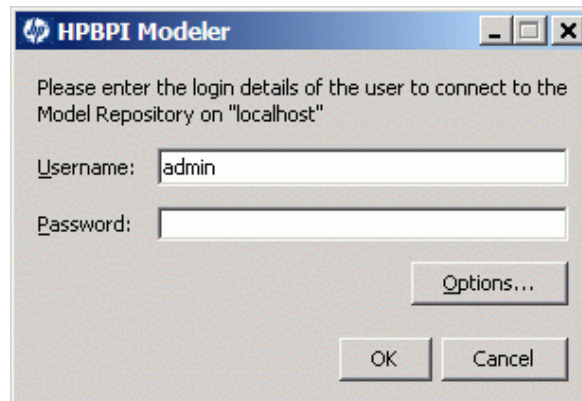
or... you can run the HPBPI Modeler executable directly, by running:

```
HPBPI-install-dir\bin\biamodeler.bat
```

Logging On

When you run the HPBPI Modeler, it prompts you for a login user name and password:

Figure 5 HPBPI Modeler Login Screen



When HPBPI is installed, a default user name and password is configured for you:

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

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

If you are running the HPBPI Modeler on a different machine to the HPBPI Model Repository, you are able to specify the hostname of the HPBPI Model Repository by clicking on the `Options...` button.

If the login fails, then the most likely causes are:

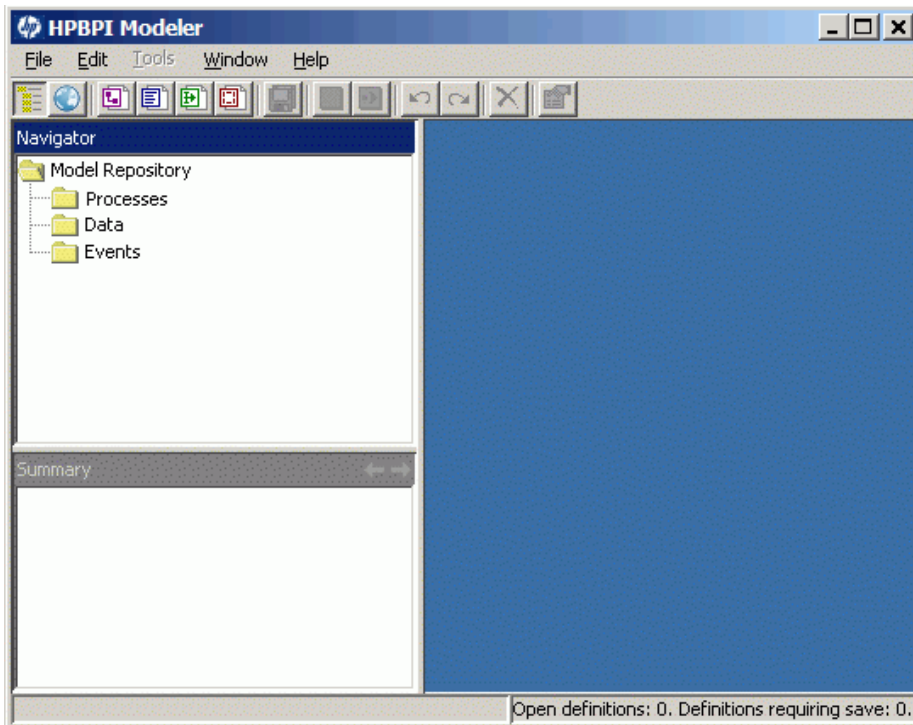
- An incorrect user name or password
- An incorrect hostname (specified under the `Options` button)
- The HPBPI Model Repository and HPBPI Servlet Engine components are not yet started on that hostname

You can use the HPBPI Administration Console, on the HPBPI server, to start the Model Repository and the Servlet Engine.

- An incorrect port number (specified under the `Options` button)
- There is a network/security reason causing the connection to fail. For example, a firewall on the server might be disallowing you to connect.

When you enter the user name and password details, and click OK, the HPBPI Modeler runs and presents a screen something like this:

Figure 6 The HPBPI Modeler



Obviously if you had previously defined some processes then these would be listed in the Navigator pane (on the left hand side).

The Model Repository

When the Modeler runs, it is connected to the HPBPI server. As you create new processes, these are saved on the HPBPI server...in the **Model Repository**.

The Model Repository is where all of your definitions are maintained. This includes your processes, all data definitions, events definitions, etc. - anything that you create in the HPBPI Modeler.

You do not actually need to know where the Model Repository is on the server, but if ever you do, it is held within the HPBPI database. The Model Repository is held in a set of data tables whose names all start with `Repos`. The main data table is: `Repos_Defns`.

Do not edit or change any of the entries in the repository data tables as you may well cause the HPBPI Modeler to crash or corrupt the repository.

Logon Users

Although you must logon to the HPBPI Modeler as a configured user, once you have logged on to the Modeler you are able to access everything within the repository.

Indeed, suppose you logon today as a user called (for example) `Fred` and create a process definition called `MyProcess`. If you logon tomorrow as a different user (for example, `admin`) you are able to continue working on `MyProcess`.

Your login user does not restrict you within the HPBPI Modeler.

Single User Repository

Although multiple developers can view the Model Repository, HPBPI currently only supports one developer actually editing the Model Repository at any one time.

If you do have multiple users writing to the HPBPI Model Repository at the same time, whoever saves their changes last overwrites the other developer's changes.

So be careful!

Drawing The Process

Now that the Modeler is running and connected to the HPBPI server, you can start to design (model) your business process.

Creating A New Process

To create a process, you can either select

File->New->Process

or simply click on the new process icon:



Create a new process...

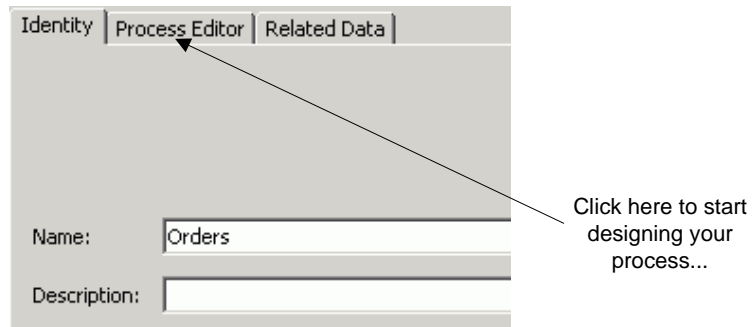
This brings up the process properties page - in the right hand pane of the HPBPI Modeler.

You can then assign the process a name and (optionally) a description. For example, you might name the process `Orders`, as follows:

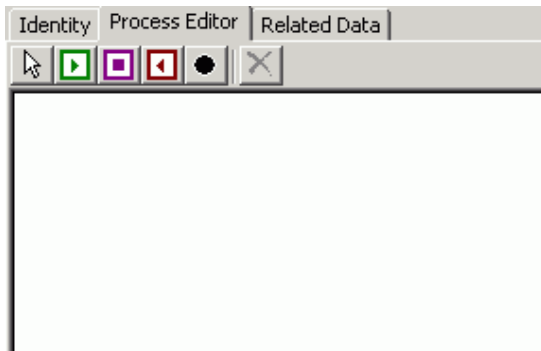
Identity	Process Editor	Related Data
Name:	Orders	
Description:		

Drawing the Steps Of The Process







Once the process is named, you can switch to the process editor pane...by clicking on the Process Editor tab, as shown:



This pane then switches to become the process editor canvas. It appears as a blank area with a selection of buttons/icons at the top, as follows:



These drawing icons perform the following functions:

	<p>Draw a Start Step</p> <p>To draw a start step you first click on this icon and then move the cursor to the desired position within the process definition pane and single click. This paints a start step at that position.</p> <p>The cursor stays in this “start step” mode, allowing you to paint more start steps. You select another icon to switch out of this mode.</p>
	<p>Draw an Activity Step</p> <p>To draw an activity step you first click on this icon and then move the cursor to desired location within the process editor pane and single click. This paints an activity step at that position.</p>
	<p>Draw a Junction Step</p> <p>To draw a junction step you first click on this icon and then move the cursor to desired location within the process editor pane and single click. This paints a junction step at that position.</p>
	<p>Draw an End Step</p> <p>To draw an end step you first click on this icon and then move the cursor to desired location within the process editor pane and single click. This paints an end step at that position.</p>
	<p>The Selection Tool Icon</p> <p>Clicking this icon switches you out of “step drawing” mode and allows you to draw arcs between steps and define step properties.</p>
	<p>Delete Steps or Arcs</p> <p>Clicking this icon deletes the currently highlighted object(s).</p>

Start Step

A start step defines where you expect this process to start.

You can have more than one start step.

Indeed, there is nothing stopping you from having no start step at all. This is because you can configure a normal activity step to start a process if that process has not already been started. So a start step is really there to help you layout your process to be as human-readable as possible.

End Step

The end step defines where you expect the process to come to an end.

You can have more than one end step.

Although you technically do not need to have an end step, you really should! The end step is used by the HPBPI Business Impact Engine to signal that a process instance has completed. Without an end step your process instances can never complete.

Junction Step

A junction step can be placed anywhere within a process diagram. You can have more than one junction step within a process diagram. You may choose to have no junction steps within your process. Junction steps are optional.

You may use a junction step to represent a step within the process that you are choosing not to monitor. You may use a junction step to represent a decision that takes place within the underlying process. Some people may find that using junction steps simply helps them layout their process diagram. Junction steps are purely there to aid with the visual layout of your process diagram.

An example of where you may see junction steps used, is when you import a Business Process Execution Language (BPEL) process into the HPBPI Modeler. In this instance, junction steps are used to help show the structure of the underlying business process being represented by the created HPBPI process diagram.

Placing The Steps For Your Process

To create a start step:

- Click on the start step icon
- Move the cursor to the location where you wish to place this start step, and single click

To create a couple of activity steps:

- Click on the activity step icon
- Move the cursor to the location where you wish to drop the first activity step...and single click
- Move to the location you wish to drop the second activity step...and single click

To create an end step:

- Click the end step icon
- Move the cursor to the location where you wish to place the end step...and single click

To **stop dropping steps** (and turn your cursor back to a normal cursor):

- Click on the selection icon (the arrow head)

For example, your process might look something like this:



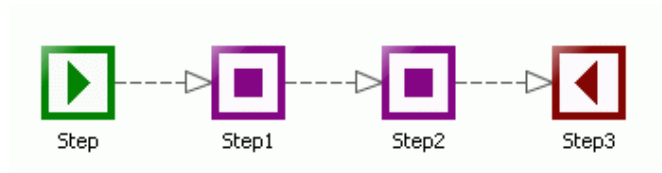
Drawing Connecting Arcs

To connect two steps (with an arc) you need to have first **clicked the selection tool icon** (arrow head).

To draw the arc, you:

- Move the cursor to be over the center of the first step
As you move the cursor over the step, the center of that step is highlighted. You must make sure that the cursor is then placed over that highlighted center part of the step. The cursor changes to a simple “cross”.
- You then left-click-and-hold the mouse
- You then drag the mouse to be over the second step
- Then let go of the mouse, and the arc is drawn

Your process might now look as follows:



Moving Step Positions

To move a step you need to have clicked the selection tool (arrow head icon).

To move a step around on the process diagram, you:

- Move the cursor to be somewhere over the step - but **not** within that center section of the step

As you move the cursor over the step, the center of that step is highlighted. You must make sure that the cursor is **not** placed over that highlighted center part of the step.

Remember, the central part of the step is used when you are wanting to draw an arc. For moving a step you aim for the edge of the step. The cursor changes to a cross with arrow heads.

- You then click-and-hold the mouse
- You then drag the step to its new location on the process editor pane.
- Then let go of the mouse, and the step has now been moved

Naming Each Step

The HPBPI Modeler assigns default names for any newly added steps.

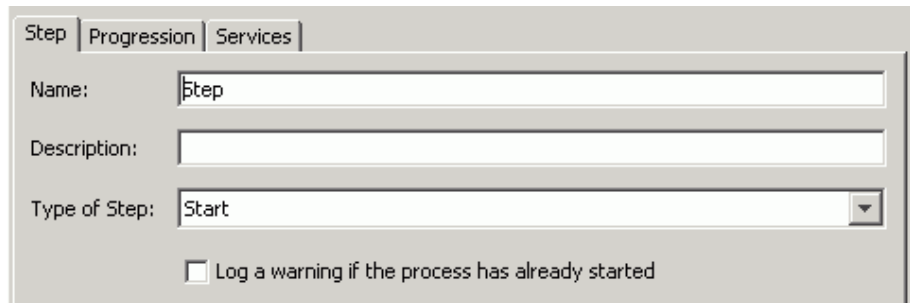
To change the name of a step you can:

- Right-click on the step and select `Properties`

...or...

- Double-click on the step

This brings up a dialog box as follows:



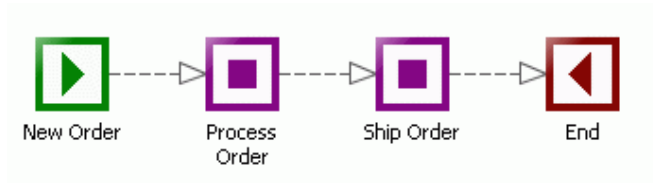
You can then enter the name of the step, and (optionally) a description.

The other tabs on this dialog allow you to set some important things such as the progression rules and service dependencies for this step.

The progression rules are where you define when a process instance has started or completed a particular step. You define these later on but, when you are drawing out each step of your process it is a good idea to be thinking about how you know when an instance has started or completed this step. That is, start to think and make notes to yourself about the conditions under which

you know that a process instance has reached this step and how the instance completes this step. You define these within the HPBPI Modeler...but not just at the moment :-)

For example, you might rename your steps of your orders process to be as follows:



Lab - Drawing The Process

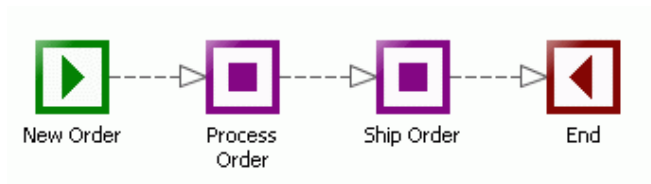
In this lab you create a process, draw some steps and then link them with arcs.

Start up HPBPI

- Run the HPBPI Administration Console
- Start up **all** the components

Start the HPBPI Modeler

- Start up the HPBPI Modeler
- Create a new process called: Orders
- Create a start step, called: New Order
- Create an activity step, called: Process Order
- Create another activity step, called: Ship Order
- Create an end step, called: End
- Now join these steps with arcs so that your process ends up looking something like this:



- Now save your work
File->Save All

or... press on the `Save All` icon in the HPBPI Modeler's main toolbar.

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

Defining The Associated Data

Once you have drawn out your process, you need to define the data properties that you want this process to maintain. Whilst it is easy to think of this data as being specific to your process instance, the HPBPI Modeler is written such that the definition of the data is actually separate from the process definition. This is done so that you can define more than one process to be driven from the one set of data properties.

Creating a New Data Definition

To create a new data definition you can

File->New->Data

or simply click on the new data icon:



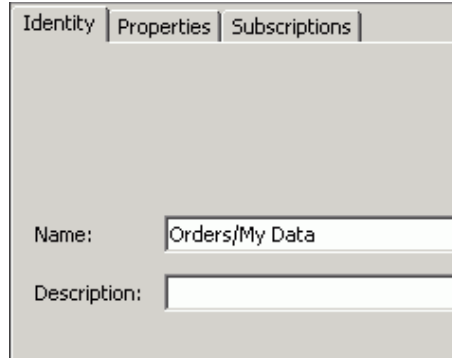
Create a new data definition..

This opens up the data definition dialog in the right-hand pane of the HPBPI Modeler.

You specify the name of this data definition and (optionally) a description.

The name of the data definition can include “/” characters to provide logical grouping. So your data definition could be called: *Orders/My Data*. This might be useful if you are planning on defining many processes all requiring data definitions, as it enables the data definitions to be listed in logical “groups”.

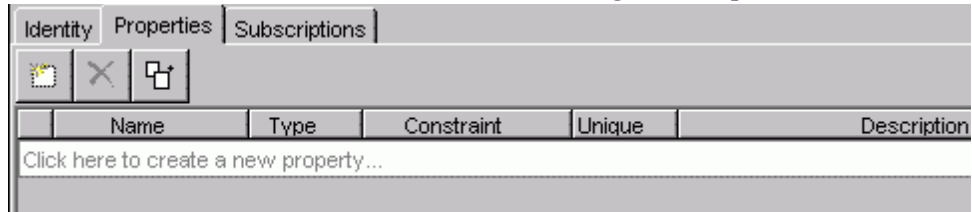
For example, you might name the data definition `Orders/My Data`, as follows:





The Data Properties

You then need to define the data properties/attributes that make up this data definition.

To do this, click on the `Properties` tab in the right-hand pane:



Where:

	<p>New Data Property</p> <p>To create a new data property you can either click in the table where it says “Click here...” or you can click this icon.</p>
	<p>Copy From Existing Property</p> <p>This icon lets you add new properties to this data definition by copying their definitions from elsewhere.</p>

For each data property you need to define the following:

- **Name**

This is the name of the property. **No spaces allowed!**

- **Type**

Here you select one of the following options:

- String

This says that the property contains a string value.

If you specify a string data type then you must also specify the maximum length that this string can be. You specify this length in the `Constraint` column for this property.

- Double

This says that the property contains a double value.

You do not need to specify any constraint for this data type.

- Integer

This says that the property contains an integer value.

You do not need to specify any constraint for this data type.

- Currency

This says that the property contains a currency (money) value. As far as the actual data inside this field, it can be any of the supported numeric types - Integer, Long or Double.

In the `Constraint` column for this property you must then specify the type of currency that this field contains. You specify the currency by typing in a valid ISO 4217 currency code. For example:

EUR - Euro
USD - American dollars
AUD - Australian dollars
GBP - British Pounds
etc...

- Date

This says that the property contains a date value.

You do not need to specify any constraint for this data type.

- Long

This says that the property contains a long value.

You do not need to specify any constraint for this data type.

- Boolean

This says that the property contains a string value containing the word “true” (in any case - upper, lower or mixed). Any other value is taken to represent “false”.

You do not need to specify any constraint for this data type.

- **Constraint**

You need to specify values in this column for the data types:

- String

- Currency

- **Unique**

You can specify that a property is to be considered unique.

You check this column if this property value is to be unique for all instances of the data definition. For example, if your data definition was holding information about specific orders then you would mark (for example) the `OrderNumber` property as being unique.

You can specify more than one property to be unique. This simply means that each of these data properties must be unique across all instances of the data. Why would you do this? It is for situations where you need to normalize your data. Let’s consider an example:

Suppose you have a process that is going to monitor an order as it moves across your business. The order is known in the `OrdersIn` system by the `OrderNumber` (for example: 12345). However, when this order moves into the `Shipping` system it is assigned a different key `OrderKey`, with a value such as ABC-XYZ. So you would need to set up your data events to start by giving you the order details according to the `OrderNumber` as defined within the `OrdersIn` system. You would then receive data events for this order and you would identify the order by this `OrderNumber` value. As the order moves into the `Shipping` system, you would need to get a data event advising you that `OrderNumber` 12345 is now assigned the `OrderKey` of ABC-XYZ. You would then begin to receive data events for the

order ABC-XYZ. In other words, your order can be identified by (in this case) any one of two unique properties. See [Multiple Unique IDs](#) on page 143 for further discussions.

- **Description**

You can enter a text string to record your comments about this property.

Deciding On The Data Properties

When defining the data properties it is difficult to get them all defined correctly up front. You would expect to define a list of properties now, and then add-to and subtract-from this list as you continue to develop the process.

When you are first drawing out the process you ask yourself:

“How do I know when the process instance gets to this step?”

Answering this question helps you to build a list of data properties.

For example:

“When the order is first written to the orders system, an order number is assigned.”

“When the shipping department assign a shipping identifier, the order has been shipped.”

These questions help you realize the following things:

- The kind of data properties you need to maintain for this process
- The kind of data events that you need to receive from the underlying application systems to enable you to monitor the process

You also need to think about the kind of statistics that you want to get from the process once it is deployed and running. For example, do you want to be able to report the customers who are placing orders or just the overall value of orders - or both - or more!

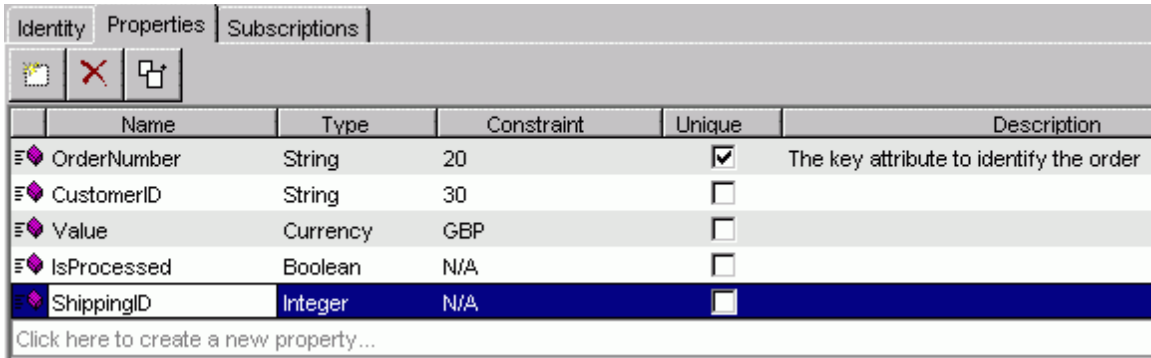
In other words, the data properties that you define provide information to enable the process to know where it is at any one time, and to collect information for later reporting/statistics.

Remember, you are only defining the data properties that you need to monitor your process - not run your real business process. That is, you do not need to define every data property from your application systems, only the important properties that you decide you wish to monitor.

Lab - Defining The Associated Data

Let's create a data definition and specify the data properties that it maintains:

- Create a new data definition, called: Orders/My Data
- Define the following properties for this data definition:



The screenshot shows a software interface with three tabs: 'Identity', 'Properties', and 'Subscriptions'. The 'Properties' tab is active. Below the tabs are three icons: a document with a plus sign, a red 'X', and a document with a plus sign. Below the icons is a table with the following columns: Name, Type, Constraint, Unique, and Description. The table contains five rows of data. The 'ShippingID' row is highlighted in blue. Below the table is a link that says 'Click here to create a new property...'

Name	Type	Constraint	Unique	Description
OrderNumber	String	20	<input checked="" type="checkbox"/>	The key attribute to identify the order
CustomerID	String	30	<input type="checkbox"/>	
Value	Currency	GBP	<input type="checkbox"/>	
IsProcessed	Boolean	N/A	<input type="checkbox"/>	
ShippingID	Integer	N/A	<input type="checkbox"/>	

[Click here to create a new property...](#)

- Make sure that you mark the `OrderNumber` as being the unique property
- Save your work

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

Relating The Data Definition To The Process

Having defined the data definition, you now need to relate this definition to the process.

To do this, you need to:

- Go to the process definition dialog

To do this you can either:

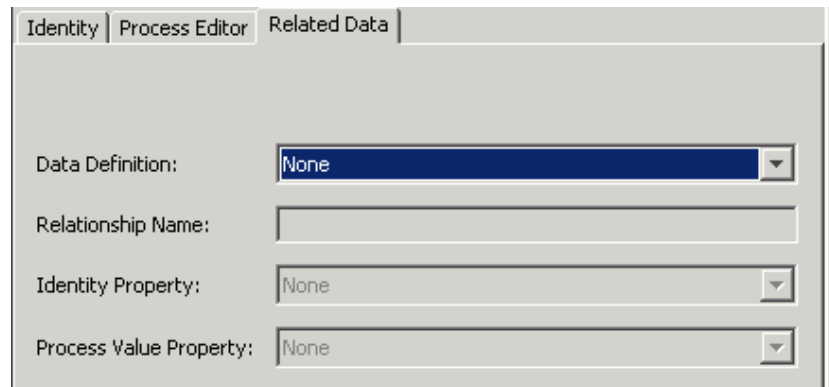
- Select `Window` and then select the process that you wish to work on

This only works if you already have the process editor open within the HPBPI Modeler.

- Or simply double-click on the name of the process in the left-hand Navigator pane

- Now click on the `Related Data` tab (in the right-hand pane)

You should see the dialog, as follows:



The screenshot shows a dialog box with four tabs: 'Identity', 'Process Editor', 'Related Data', and an unlabeled tab. The 'Related Data' tab is selected. The dialog contains four fields:

- Data Definition:** A dropdown menu with 'None' selected.
- Relationship Name:** A text input field.
- Identity Property:** A dropdown menu with 'None' selected.
- Process Value Property:** A dropdown menu with 'None' selected.

where:

- **Data Definition**

You open the selection box and select the data definition that you wish to relate to this process.

- **Relationship Name**

This allows you to specify a name for this relationship. It is actually not very important so you may as well accept the default value specified.

- **Identity Property**

You **must** assign an identity.

This is where you choose one of the properties from this data definition to be marked as the identifier. You can choose any property you like.

Choosing a property to be the Identity Property is only of importance when you come to run the HPBPI Business Process Dashboard to see your running processes, once everything is deployed and running. It is the property shown with each process instance to identify it on the screen.

Although you can choose any property you wish as the Identity, you typically choose a property that is marked as unique.

- **Process Value Property** (optional)

The process value property is like the Identity property in that it is only used when you report on the process. The HPBPI Business Process Dashboard displays the process value property with each process instance.

To set the process value property, you choose one of your numeric data properties from the related data object.

The process value property is for numeric data only.

Non-numeric properties are grayed-out (not available) in the pull-down list of choices.

For example, you might set up the related data as follows:

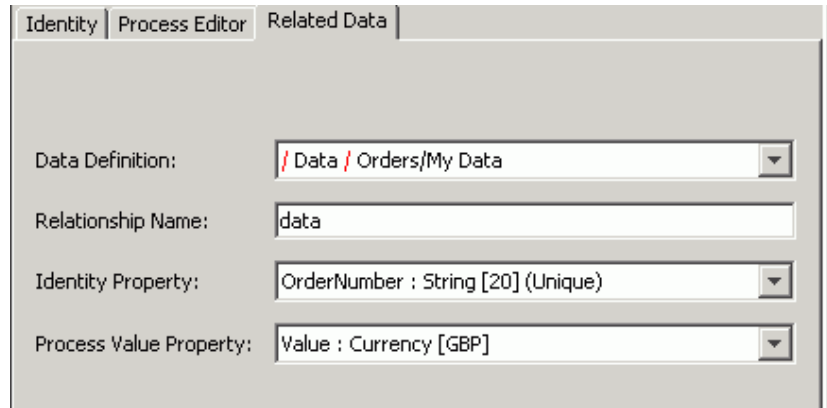
The screenshot shows a configuration window titled 'Related Data' with the following fields:

- Data Definition:** / Data / Orders/My Data
- Relationship Name:** data
- Identity Property:** OrderNumber : String [20] (Unique)
- Process Value Property:** Value : Currency [GBP]

Lab - Relating The Data Definition

Let's relate the `Orders/My Data` definition to the process `Orders`.

- Double click on `Orders` in the Navigator pane
- Select the `Related Data` tab
- Configure the relationship to be as follows:



The screenshot shows a dialog box titled "Related Data" with three tabs: "Identity", "Process Editor", and "Related Data". The "Related Data" tab is active. It contains four fields:

- Data Definition:** A dropdown menu showing "/ Data / Orders/My Data".
- Relationship Name:** A text box containing "data".
- Identity Property:** A dropdown menu showing "OrderNumber : String [20] (Unique)".
- Process Value Property:** A dropdown menu showing "Value : Currency [GBP]".

Note: The dialog displays the data definition name with a `/Data/` prefix just to highlight the fact that it is a data definition :-)

- Save your work

At this point in your process definition, you have:

- Drawn the process
- Defined the data
- Related the data to the process

The next step is to configure the **progression rules** for this process...

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

Configuring Progression Rules

Now that the process has an associated (related) data definition, you can go through and configure the progression rules for each step in the process.

Progression rules are often called **start** and **complete conditions** (or start and complete criteria) because they configure the rules to say whether a step has been started or completed.

Progression rules are based solely around the values within the data definition. For example:

- This step is started when the `ShippingID` property is assigned a value for the first time
- This step is started when a `Claims_Handler_ID` has been assigned and there is a valid `Authorization_Check` and you have not yet assigned the `Claim_Value`
- This step is completed when the `CreditCheck` property is assigned `true`

Progression rules can be simple or complex. It all depends on the process you are monitoring.

To define the progression rules for a step, you need to:

1. Display the process within the right-hand pane of the HPBPI Modeler

You can achieve this by double-clicking on the process (in the left-hand Navigator pane) and then selecting the `Process Editor` tab.

2. Then double-click on the step

Or...right-click and select `Properties`.

This brings up the **Step Properties** dialog - the one you used earlier to name the steps.

3. Within this Step Properties dialog, click on the **Progression** tab

The Rules Language

There is actually a language for writing progression rules and the grammar for this language is defined in the *HPBPI Reference Guide*.

There are also some easier ways to enter progression rules...

Style Of Progression

When developing business processes there are some common subsets of the progression rule language that people often use. These “simpler” forms of progression are available within the HPBPI Modeler as menu driven options. For more complex progression rules, you can choose to enter the progression rule by hand. So, when configuring progression rules for a step you choose your *style* of progression.

The choice of styles is as follows:

- **Complete on first assignment**

With this style of progression you simply select the data property (from the pull down list) that you wish to monitor. The first time that this property is assigned a value, this step is marked as completed.

- **Complete on transition**

This style lets you select a data property from the pull down list, and then you can specify the transition that you want to catch.

You can specify the `From` and `To` values.

If the property is defined as a `String` property then you need to specify the string value within double quotes (“string value”).

You can use the keyword `null` in either the `From` or `To` fields.

Here are some examples:

Example 1:

```
Property: MyOrder
From:     null
To:
```

Specifies that you are looking for when `MyOrder` makes the transition from being a `null` value to being any other value.

Example 2:

```
Property: MyOrder
From:
To:       "Received"
```

Specifies that you are looking for when `MyOrder` makes the transition from any value to the value `"Received"`.

Example 4:

```
Property: MyOrder
From:     "Received"
To:       "Processed", "Fixed", "Solved"
```

Specifies that you are looking for when `MyOrder` makes the transition from the string value `"Received"` to one of the three specified string values `"Processed"`, `"Fixed"` or `"Solved"`.

- **Start and complete on transition**

This style lets you configure separate start and completion transitions.

You select a data property to signal when this step starts, and a data property to signal when this step completes. This can be the same data property or a different one.

Here are some examples:

Example 1:

```
Start transition:  
  Property: MyOrder  
  From:     null  
  To:
```

```
Complete transition:  
  Property: ShipID  
  From:     null  
  To:
```

Specifies that:

- This step starts when `MyOrder` transitions from `null` to any value.
- This step completes when `ShipID` transitions from `null` to any value.

Example 2:

```
Start transition:  
  Property: Priority  
  From:  
  To:      1
```

```
Complete transition:  
  Property: Priority  
  From:  
  To:      5,6,7,8
```

Specifies that:

- This step starts when `Priority` transitions from any value to the value 1.
- This step completes when `Priority` transitions from any value, to any one of the values 5, 6, 7 or 8.

- **Advanced conditions**

If your progression rules need to be more complex than the other styles available, you can select this mode and you can enter your start and completion conditions using the language as described later in [Chapter 5, Advanced Progression Rules](#).

For example, you would need to use the `Advanced conditions` option if your step start (or completion) condition was dependent upon values in two or more data properties.

Switching Styles

You can always start to define your progression rules using one of the simpler styles, and then switch to `Advanced conditions` to see what this actually translates to in the underlying progression rule language. You can then switch back.

When you try to switch to a progression style, the HPBPI Modeler first checks that your rules can be expressed in that style. If not, then you are warned that you could lose your progression rules if you continue with this switch. You can then choose to continue, or to cancel and leave the progression rules as they are.

No Implied Step Sequence

When defining a completion condition for a step it is sometimes easy to assume that when this completion condition is satisfied (at run time) that the process instance automatically moves onto the next step. This is not true.

The fact that your process diagram shows arcs from one step to the next does not guarantee any sequence or processing order. If a step's completion condition is met then that step completes. But that's it. It only enters the next step as and when that step's start condition is satisfied.

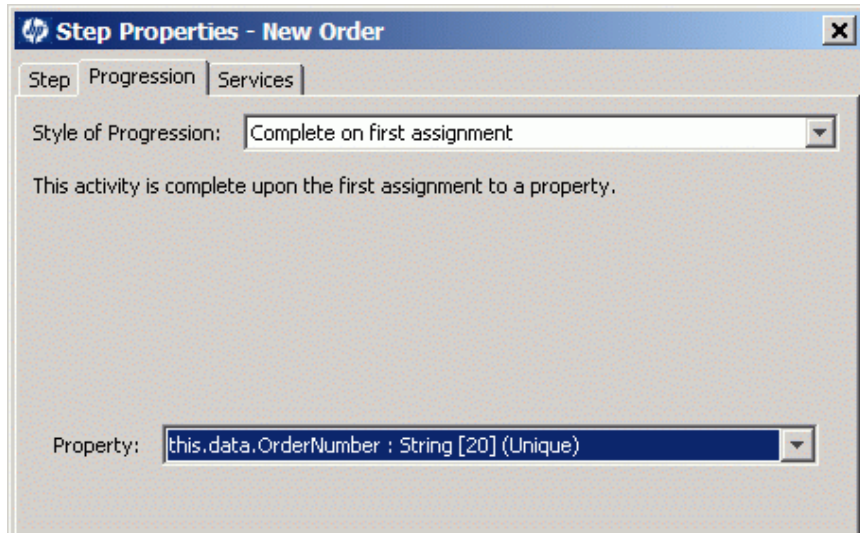
This is something that you need to think carefully about when defining your process.

If you want to ensure that the process instance moves from one step to the next then you might want to make sure that the start condition for the second step is the same as the completion condition for the step.

Lab - Progression Rules

Let's configure the progression rules for your `Orders` process:

- Display the `Orders` process in the right-hand pane and click on the `Process Editor` tab
- Configure the `New Order` step progression rule as follow:



Every time a new order number comes into the HPBPI Business Impact Engine, this creates a new instance of the data definition and assigns a value to the `OrderNumber` property. The above progression rule causes a new process instance to be allocated and the `New Order` step is marked as completed.

- Configure the `Process Order` step progression rules as follow:

The screenshot shows a dialog box titled "Step Properties - Process Order" with three tabs: "Step", "Progression", and "Services". The "Progression" tab is active. It features a dropdown menu for "Style of Progression" set to "Start and complete on transitions". Below this, a note states: "This activity starts and completes on separate property transitions." The "Start Transition" section includes a "Property" dropdown set to "this.data.OrderNumber : String [20] (Unique)", a "From" text box containing "null", and an empty "To" text box. The "Complete Transition" section includes a "Property" dropdown set to "this.data.IsProcessed : Boolean", an empty "From" text box, and a "To" text box containing "true".

This step starts when the `OrderNumber` property is assigned a value (other than null).

This step completes when the `IsProcessed` property changes to the value `true`.

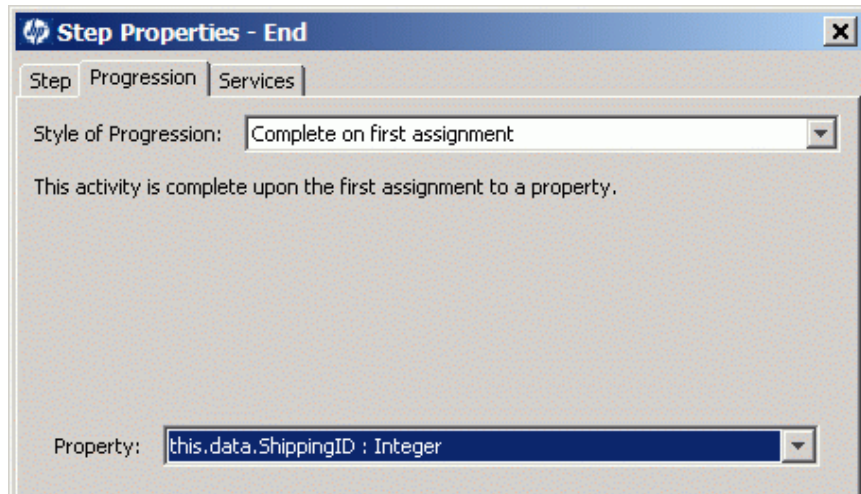
- Configure the `Ship Order` step progression rules as follow:

The screenshot shows a dialog box titled "Step Properties - Ship Order" with three tabs: "Step", "Progression", and "Services". The "Progression" tab is active. It features a dropdown menu for "Style of Progression" set to "Start and complete on transitions". Below this, a note states: "This activity starts and completes on separate property transitions." The "Start Transition" section includes a "Property" dropdown set to "this.data.IsProcessed : Boolean", an empty "From" text box, and a "To" text box containing "true". The "Complete Transition" section includes a "Property" dropdown set to "this.data.ShippingID : Integer", a "From" text box containing "null", and an empty "To" text box.

This step starts when the `IsProcessed` property changes to the value `true`.

This step completes when the `ShippingID` property changes from `null` to something. (In other words, when the `ShippingID` property is assigned a value for the first time.)

- Configure the `End` step progression rule as follow:



This step completes when the `ShippingID` property is first assigned a value.

- Save your work
- Click on the `New Order` step and bring up the step properties dialog
- On this step properties dialog, select the `Progression` tab
- Leave this dialog box showing on the screen, and back on the actual process diagram, click on the `Process Order` step

Notice that the step properties dialog has now switched to represent the currently selected step.

So, when developing a process, you can keep the step properties dialog open while you move between the steps. There is no need to close down the step properties dialog.

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

Defining The Data Events

So far you have:

- Drawn the process
- Defined the data
- Related the data to the process
- Defined the step progression rules

You now need to define the **data events** that are needed to drive this process...

Define Each Event

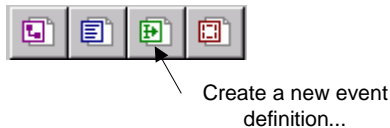
You first need to define the actual events that you expect to come from the underlying application systems.

Remember that defining these events here in the HPBPI Modeler does not mean that the application systems are set up to generate them. That is a separate step required later on when you go to run this process for real. Refer to the *HPBPI Integration Training Guide - Business Events* for more details on how to do this.

To create an event definition, you can either select:

File->New->Event

or simply click on the new event icon:

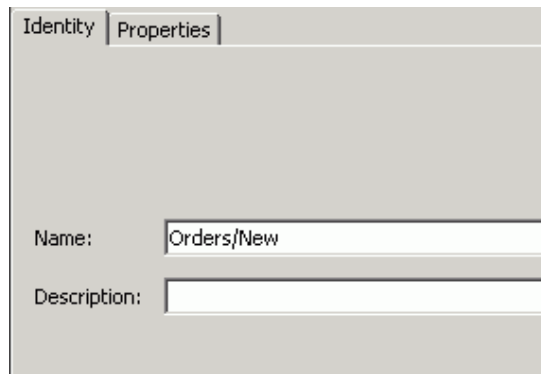


This brings up the event properties page - in the right hand pane of the HPBPI Modeler.

You specify the name of this event definition and (optionally) a description.

The name of the event definition can include “/” characters to provide logical grouping. So your event definition could be called: `Orders/New`. This might be useful if you are planning on defining many processes all requiring event definitions, as it enables the event definitions to be listed in logical “groups”.

For example, you might name your event definition `Orders/New`, as follows:

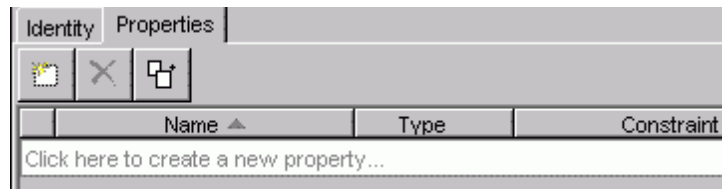


Identity | Properties




Name:

Description:

You then click on the `Properties` tab and you should see a dialog similar to this:



Identity | Properties

Name ▲	Type	Constraint
Click here to create a new property...		

You can go through and manually create new properties for this event by clicking where it tells you to ...however... by using the `Copy Properties` icon:



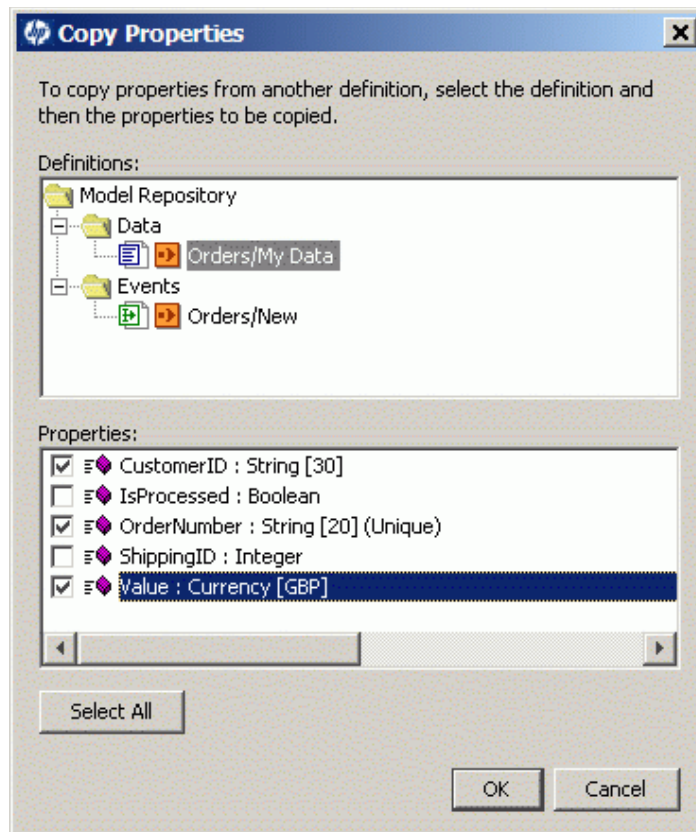
you can select properties from an existing data definition and the HPBPI Modeler creates properties to match their name and definition.

This is really helpful because your events are bringing in data that you typically want to store into a data definition. So it saves you having to type the same information over and over.







When you click on the copy properties icon, you are presented with a dialog. You then:

- Select the definition from which you wish to copy
- Select the properties within that definition that you wish to actually copy to this event definition

For example, the Copy Properties dialog might look something like this:



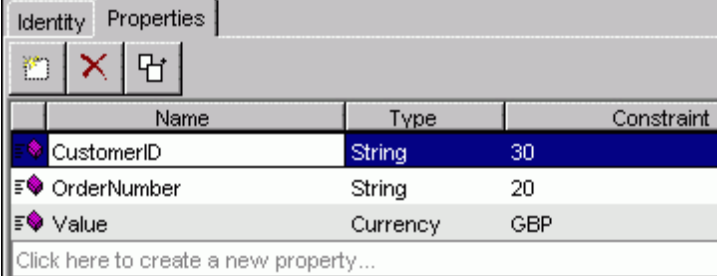
You press OK and it adds these properties to your event definition.

Identity		Properties	
			
	Name	Type	Constraint
	CustomerID	String	30
	OrderNumber	String	20
	Value	Currency	GBP
Click here to create a new property...			

Lab - The Data Events

Let's define the events for the `Orders` process

- Create a new event, and call it: `Orders/New`
- Click on the `Properties` tab for this event
- Using the `Copy properties...` icon, define this event to have the following properties:



Name	Type	Constraint
CustomerID	String	30
OrderNumber	String	20
Value	Currency	GBP

Click here to create a new property...

- Create two more events as follows:

Event: `Orders/Processed`

Properties:
`OrderNumber`

Event: `Orders/Shipped`

Properties:
`OrderNumber`
`ShippingID`

- Save your work

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

Configuring The Event Subscriptions

Having defined the events, you have specified what data you expect to receive from the underlying application systems. You now need to configure how these events are going to update the `Orders/My Data` data definition.

You might be thinking:

“But I’ve defined these events specifically for this process and data definition?”

But you might use these events to update other data definitions in the future. You see, the HPBPI Modeler allows you to define your data, your events and your processes all as separate entities. It is then up to you as to how you piece them together.

Anyway, in this case, you do want to configure these events such that they update your `Orders/My Data` definition.

The way you do this is by configuring the data definition to **subscribe** to these events.

Setting Up A Subscription

A subscription is something that you set up on the data definition:

- Double click on the data definition (for example: Orders/My Data)
- In the right-hand pane, click on the Subscriptions tab
- Click to create a new subscription

An event subscription dialog similar to the following then appears:

Event Subscription - Orders/New [X]

Event:

Description:

Associate with a specific instance

Event is for a specific instance of this data definition.

Unique Property of this Data Definition: == Property of Event:

In order to handle the event, should the data instance already exist?

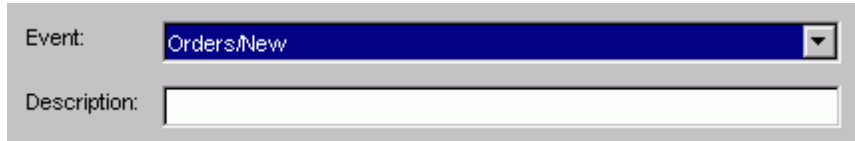
Filter events or data definitions

Actions

Terminate the instance of the data definition after handling the event.

This event subscription dialog looks complicated but is actually quite straightforward. The dialog actually consists of the following four main parts:

Event and Description



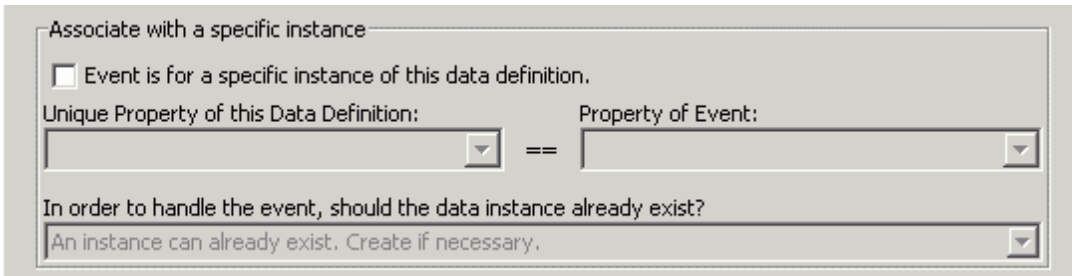
Event: Orders/New

Description:

Here you select the event, from the available list, to which you wish to subscribe.

You can optionally enter a description.

Specific or Not



Associate with a specific instance

Event is for a specific instance of this data definition.

Unique Property of this Data Definition: == Property of Event:

In order to handle the event, should the data instance already exist?

An instance can already exist. Create if necessary.

Here you specify whether the event (that you are subscribing to) is destined for a specific data instance or for all instances of this data definition.

For example, you could specify that this event is for a specific instance of your data, and then specify which property in the event is to be matched with the data definition.

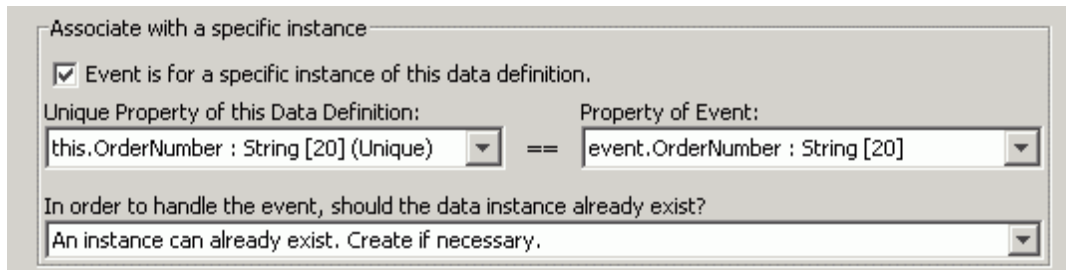
Or, you might not check the box and thus this event affects **all** instances of your data definition. This allows an event to update all the data instances with information.

At first, you are likely to set up events for specific instances. So typically you would check the box to say it is for a specific data instance, and then specify the event property whose value is to be used to match up with the unique property within the data definition.

If you have specified that this event is for a specific data instance, you then select the behavior for when the event comes in and there is currently no specific data instance for this event. You have a series of options available to you where you can ask that the data instance be created (this is typically the option to select), ask for it to create the data instance but to log a warning, not create the data instance, retry the event later, etc.

The typical option is that you want the data instance to be automatically created the first time.

For example:



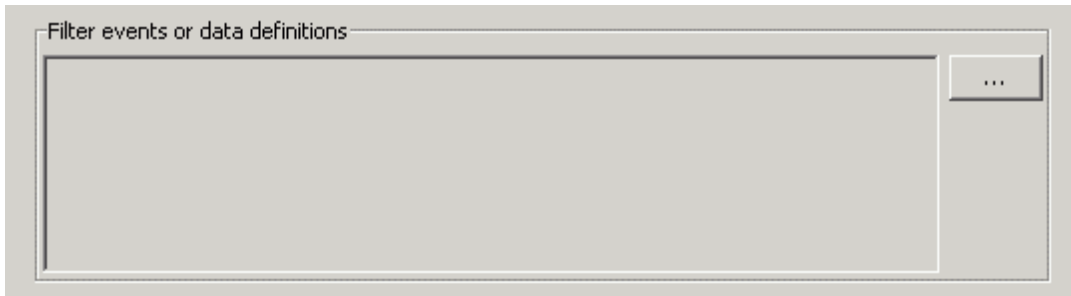
The screenshot shows a configuration dialog box with the following elements:

- Title: Associate with a specific instance
- Checkbox: Event is for a specific instance of this data definition.
- Unique Property of this Data Definition: `this.OrderNumber : String [20] (Unique)`
- Property of Event: `event.OrderNumber : String [20]`
- Comparison operator: `==`
- Question: In order to handle the event, should the data instance already exist?
- Answer: An instance can already exist. Create if necessary.

where:

- This event subscription is for a specific data instance
- Match the incoming event to the data instance by matching the data instance's `OrderNumber` property (`this.OrderNumber`) with the incoming event's `OrderNumber` property (`event.OrderNumber`)
- If there is no current data instance whose `OrderNumber` matches the incoming event, create a new data instance

Filtering



By clicking on the “ ... ” button, you can set up additional filtering to be applied to the incoming event. This filtering can be any expression that results in a boolean `true` or `false`.

Example 1:

```
event.CustomerType == "Gold"
```

This means that the incoming event is only accepted if the event property `CustomerType` is set to the string `Gold`.

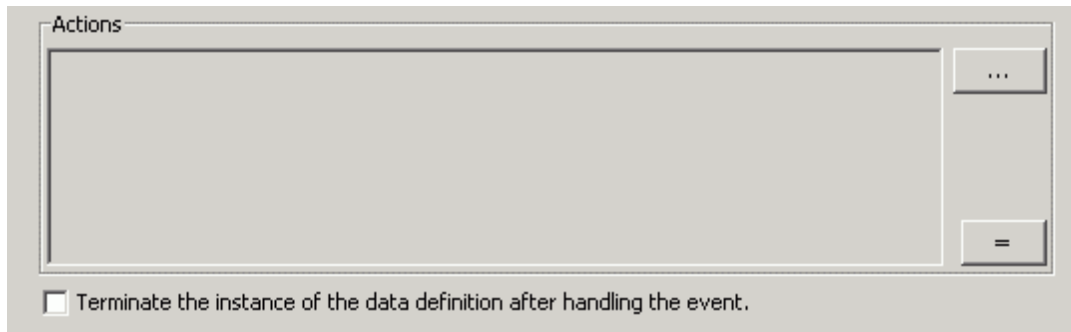
Example 2:

```
event.CustomerType == "Silver"  
&& event.Value > 1000
```

This means that the incoming event is only accepted if the event property `CustomerType` is set to the string `Silver` and the event property `Value` is set to a value greater than `1000`.

Your filters can be based on the property values within the incoming event and/or the current data instance values.

Actions



The Actions section is where you specify which properties are pulled out of the event and placed into the data definition.

The basic syntax for these actions is:

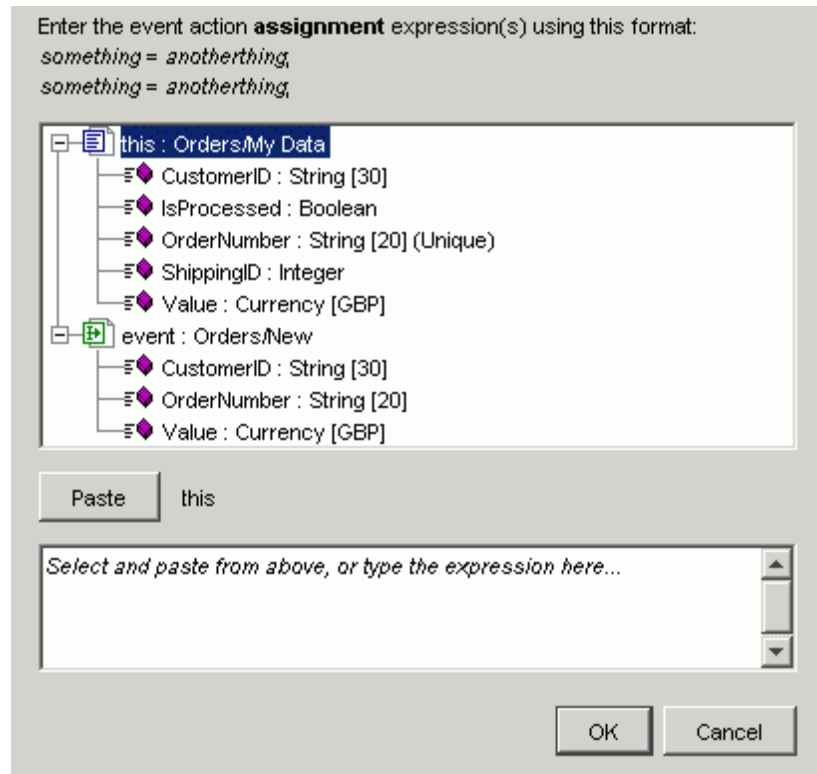
```
this.Property1 = event.EventPropertyA;  
this.Property2 = event.EventPropertyB;  
...
```

You are specifying which event properties to assign to data instance properties.

The “ = ” button is the quick way to simply say:

“Set up assignments for each property in the event that has a matching named and typed property within the data definition.”

If you wish to be more selective in your assignments then you can press the “...” button. This presents a dialog where you can select the properties that you require. This dialog looks something like this:



where:

- You are presented with property lists for both the data definition and the event definition
- You can then either type the assignments in at the bottom of the screen yourself ...or...
 - Select the data property
 - Click the `Paste` button
 - Type in =
 - Select the event property

- Click the Paste button
- Type in ;
- When you have your set of assignments...click OK

Ternary Expressions

You can also use ternary expressions within the actions. A ternary expression is a shortened form of an `if` statement, for example:

```
this.dataAtt = (condition) ? true_result : false_result ;
```

Using the ternary expression allows you to assign different values based on the result of an `if` condition.

For example:

```
this.code = (event.MyType == "B") ? "Assigned" : "NoGood";
```

This sets the `code` data attribute to the string `Assigned` if the attribute `MyType` (within the incoming event) is set to `B`. If `MyType` is not equal to `B` then `code` is set to the string `NoGood`.

```
this.b_state = (event.MyType == "B") ? true : this.b_state;
```

This sets the `b_state` data attribute to the boolean value `true` if the `MyType` attribute (within the incoming event) is set to the string value `B`. If the `MyType` attribute is not set to `B` then `b_state` is set to its current value...in other words...the value of `b_state` is left untouched.

Mathematical Expressions

You can also carry out mathematical calculations, for example:

```
this.Amount = this.Amount + 1;
```

You can use `+`, `-`, `*`, `/` and you can use brackets `()` to help with the precedence.

Some examples:

```
this.Amount = this.Amount * 2;
```

This means that when the event arrives, the value in the data attribute `Amount` is multiplied by 2.

```
this.Amount= (event.Prop1 > 10) ?  
    this.Amount / 2  
    : this.Amount;
```

This means that if the value in the `Prop1` attribute of the incoming event has a value greater than 10, then the `Amount` attribute value is divided by 2...otherwise the `Amount` attribute value remains unchanged.

```
this.Amount= (event.Prop1 > 0) ?  
    ((this.Amount+13)/2)*(this.Amount/event.Prop1)  
    : this.Amount;
```

This means that if the value in the `Prop1` attribute of the incoming event has a value greater than zero then the calculation is applied...otherwise the `Amount` attribute remains unchanged.

Type coercion

The assignments you specify in your actions can include type coercion. That is, you can (for example) assign an integer property to a string. The run time HPBPI Business Impact Engine is able to handle basic type coercion as described in the *HPBPI Reference Guide*.

Action Order and Errors

The actions you specify are carried out in the order you specify. This is (possibly) important in the case where an error occurs. If an error occurs on an assignment then that action fails and all subsequent actions are not applied. However, all previous actions have been assigned successfully.

A possible error would be if you were assigning a string to an integer and the string happened to contain non-numeric characters.

Terminating The Data Instance

You check this box to terminate the data instance after handling this event. You would typically check this box for the event that signalled the end of the process instance. That is, when the event arrives to end this process instance, you want it to also terminate the data instance.

Be careful if your data instance is being shared between more than one process. In this situation you would need to make sure that you did not terminate the data instance prematurely.

If you do not have an event subscription that terminates the data instance then the data instance remains in the HPBPI database marked as *Active*.



You should always have at least one event subscription that terminates the data instance.

Lab - Event Subscriptions

Let's configure the subscriptions for your `Orders` process:

- Double-click the `Orders/My Data` data object (in the Navigator pane)
- Click on the `Subscriptions` tab (in the right-hand pane)
- Click to create a new subscription

Orders/New

When the subscriptions dialog shows:

- Select the `Orders/New` event from the pull-down
- Check the box to say this subscription is specific for a data instance

This pre-fills the boxes and because you have the `OrderNumber` defined in both the data and event definitions these are automatically configured for you.

- In the Actions section, click the “ = ” button and this sets up the necessary assignments

Again, because you have the same named properties in both the data and event definitions the HPBPI Modeler is able to automatically configure these actions for you.

Your subscription should look something like this:

Event Subscription - Orders/New

Event: Orders/New

Description:

Associate with a specific instance

Event is for a specific instance of this data definition.

Unique Property of this Data Definition: this.OrderNumber : String [20] (Unique) == Property of Event: event.OrderNumber : String [20]

In order to handle the event, should the data instance already exist?
An instance can already exist. Create if necessary.

Filter events or data definitions

Actions

```
this.OrderNumber = event.OrderNumber;  
this.CustomerID = event.CustomerID;  
this.Value = event.Value;
```

Terminate the instance of the data definition after handling the event.

OK Cancel

- Click OK

Orders/Processed

For this subscription you do the same as before:

- New subscription
- For `Orders/Processed` event
- Specific for a data instance
- Click the “ = ” button

However, the “ = ” button only assigns the `OrderNumber` property and you also want this subscription to assign the data instance’s `IsProcessed` property.

This is a case where the event itself is enough to signal that the order has been processed. So just the fact that you have received this event means that the subscription can assign the `IsProcessed` property to `true`. There is no need to receive this property from the actual event - and thus no need for the underlying application systems to generate the `IsProcessed` property.

So, to add this assignment to your event:

- Click on the “...” button (for the `Actions` section)
- In the text at the bottom of this dialog - the area that currently contains the assignment `this.OrderNumber = event.OrderNumber;`

Append a carriage return after the “;” at the end of the first line. In other words, start a second line.

- Now click on the `IsProcessed` property (within the `this: Orders/My Data` section) near the top of the dialog
- Click on the `Paste` button

You should see the text `this.IsProcessed` appear in the text area at the bottom of the dialog.

- In the text area near the bottom of the screen, click at the end of the line that has just appeared, and type in: `= true;`
- Click OK

Your subscription should now look something like this:

Event Subscription - Orders/Processed

Event:

Description:

Associate with a specific instance

Event is for a specific instance of this data definition.

Unique Property of this Data Definition: ==

Property of Event:

In order to handle the event, should the data instance already exist?

Filter events or data definitions

...

Actions

```
this.OrderNumber = event.OrderNumber;
this.IsProcessed = true;
```

...

=

Terminate the instance of the data definition after handling the event.

OK Cancel

- Click OK

Orders/Shipped

- The Orders/Shipped event completes the process instance. You want to mark that this subscription also terminates the data instance
- Go ahead and create the following subscription:

Event Subscription - Orders/Shipped

Event:

Description:

Associate with a specific instance

Event is for a specific instance of this data definition.

Unique Property of this Data Definition: == Property of Event:

In order to handle the event, should the data instance already exist?

Filter events or data definitions

...

Actions

```
this.OrderNumber = event.OrderNumber;  
this.ShippingID = event.ShippingID;
```

...

=

Terminate the instance of the data definition after handling the event.

OK Cancel

Don't forget to check the box to say that the instance of the data definition is terminated after the event is handled.

- Save your work

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

Summary

Let's summarize the steps (so far) to model a process:

- 1. Understand the real business process**

In the real world you would have spent time gaining an understanding of the actual business process and produced from this a high-level version of this business process that represents the main phases.

During this phase it is important to ask yourself the question:

“What statistics do I want to get from this?”

You would also want to list out the possible sources for data events for each step within your high-level process.

- 2. Draw the high-level process**

Run the HPBPI Modeler and layout the high-level process diagram, assigning meaningful names to each step.

- 3. Define the data definition**

Define the data properties/attributes that are to be maintained.

- 4. Relate the data definition to the process**

- 5. Configure the progression rules for each step**

These allow the process to know where it is at any given time.

- 6. Define the data events**

Define the expected data to be received from the underlying application systems.

- 7. Configure the event subscriptions**

This configures how the application data is assigned/mapped to the data instance.

At this stage you have a process that is ready to be deployed to the HPBPI Business Impact Engine.

Deploying the Process

Before you can deploy a process you need to make sure that the HPBPI Business Impact Engine is up and running. You can use the HPBPI Administration Console to check this.

You deploy a process from within the HPBPI Modeler.

You simply:

- Select the process name in the Navigator pane
- Then either
 - Click the green deploy icon



- Or select: **File->Deploy...**

This then deploys the selected process and any associated definitions that have changed since the previous deploy (if any).

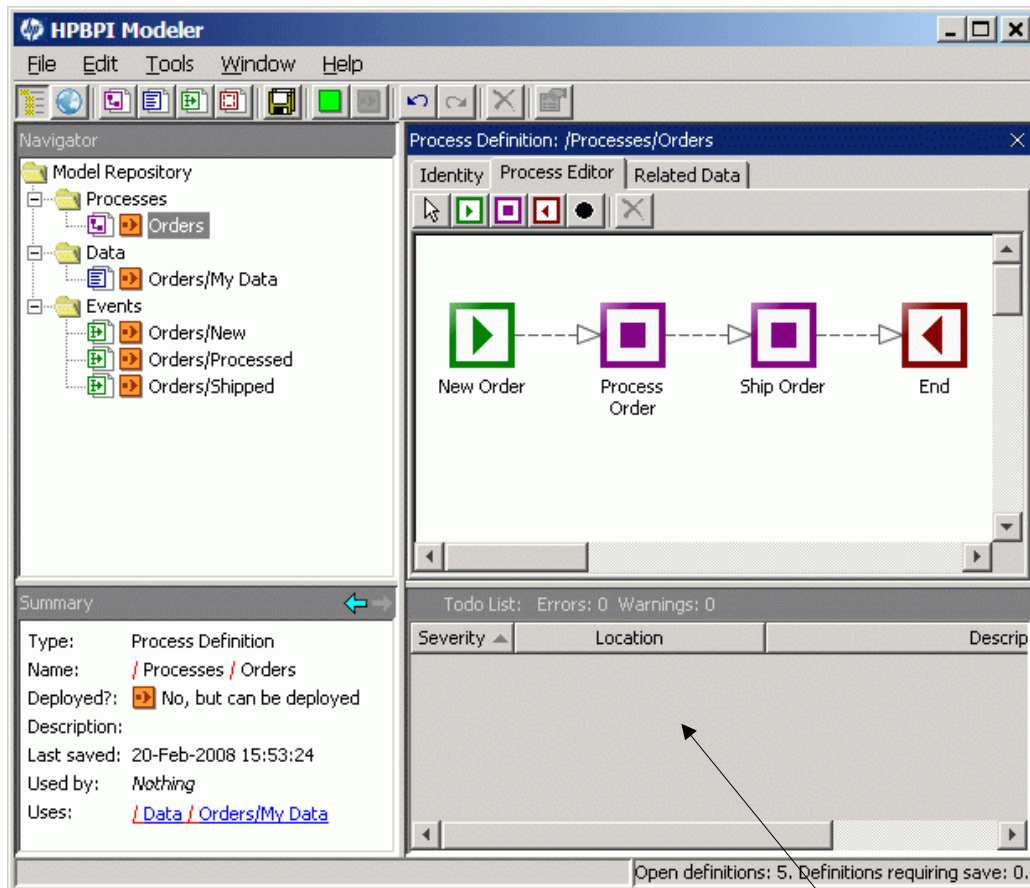
- You see a pop-up dialog listing the definitions that are to be deployed and you can then click OK (or cancel)

The ToDo List

Before deploying your process it is always worth checking that there are no warnings or errors listed in the ToDo list.

The ToDo list is displayed in the right-hand pane of the HPBPI Modeler, underneath the current definition pane.

For example:



The ToDo list

If you cannot see any ToDo list area then it might be set to “zero height”, in which case you simply need to expand it by using the mouse to enlarge the window.

Viewing the Process

“Once a process is deployed within HPBPI, how can I see it?”

To view the deployed process you can use the **HPBPI Business Process Dashboard** that comes standard with the product. The Business Process Dashboard allows you to view any process and see basic statistics about it.

The Business Process Dashboard URL

For the HPBPI Business Process Dashboard to be able to work, your HPBPI administrator must have first started the **Servlet Engine** component on the HPBPI server. This is because the HPBPI Business Process Dashboard is a set of JSP pages that require a servlet engine to run them.

To run the HPBPI Business Process Dashboard you can do one of the following:

- Start a Web browser and type in the URL:

```
http://hostname:44080/hpbpi-bpd
```

where:

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

For example:

```
http://localhost:44080/hpbpi-bpd
```

This works if your HPBPI Business Impact Engine is installed on the same machine as your Web browser.

...or...

- Select

```
Start->Programs->HP->HP Business Process Insight->Dashboard
```

Logging On

When HPBPI is first installed there is no logon required to use the HPBPI Business Process Dashboard. You can simply run the HPBPI Business Process Dashboard and see all of your processes and their data.

The HPBPI Administration Console allows you to specify whether you require people to give a username and password when running the HPBPI Business Process Dashboard. (See the *HPBPI System Administration* guide for more details.)

Testing the Process

Q: “Now the process is deployed, is there any way to test it out?”

A: “Yes!”

Obviously, for your process to go live, someone has to work with the applications people to generate the required events that then drive the process. This can be quite involved and it may be that your applications people are unable to generate the events exactly as you have defined, in which case you would then have to go back and alter your HPBPI process definition. For more details on how to configure the actual events from the underlying application systems you can refer to the *HPBPI Integration Training Guide - Business Events*.

But it would be good to be able to drive the process without having to invest time developing links to the application systems. This can help you spot any errors in your progression rule logic. It can also help you demonstrate the process to your management/customer base to get further feedback on whether the process is right for your needs.

To test out your business process you can use the **Process Simulator** contributed utility.

The Process Simulator 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.

To run the Process Simulator, you just need to double-click the `ProcessSimulator.bat` file in the `HPBPI-install-dir\contrib\ProcessSimulator` directory

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

Lab - Testing the Orders Process

Let's deploy the `Orders` process and then test it out by sending in some data events and seeing what happens...

Deploying the Process

- Make sure that the `Business Impact Engine` component is up and running on the HPBPI server
- Using the HPBPI Modeler, deploy the process: `Orders`

You should see that it deploys:

- The three data events
- The data definition
- The orders process

Viewing the Deployed Process (HPBPI Dashboard)

- Make sure that the `Servlet Engine` component is up and running on the HPBPI server
- Select:
Start->Programs->HP->HP Business Process Insight->Dashboard
- Click on the process name and it should then show you the `Orders` process diagram.

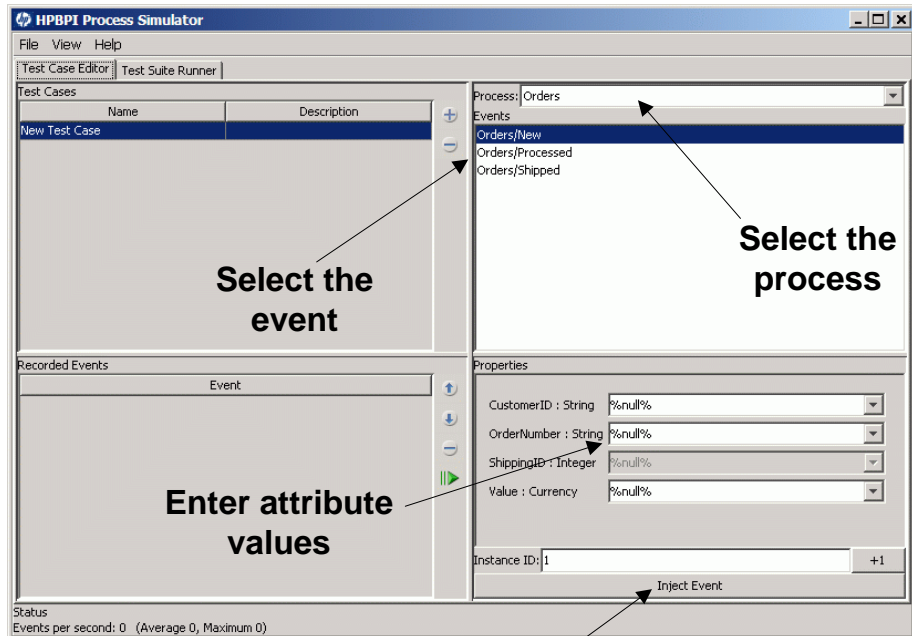
This details page also shows that there are currently no active instances.

Let's create some process instances...

Generating Data Events (Process Simulator)

- Make sure that the Business Event Handler component is up and running on your HPBPI server
- cd to the directory `HPBPI-install-dir/contrib/ProcessSimulator`
- Double-click the file `ProcessSimulator.bat`

The Process Simulator runs and a screen appears that looks like something like this:



- Select the process `Orders`
- Select the event `Orders/New`
- Enter the following event attribute values:

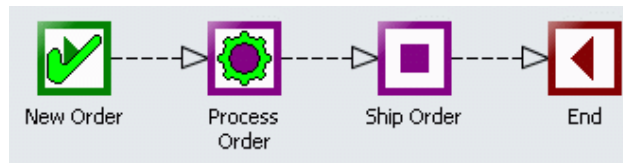
```
CustomerID = Cust1  
OrderNumber = Ord1  
Value = 10.99
```
- Click the `Inject Event` button at the bottom of the screen.

Back in the HPBPI Business Process Dashboard

In the HPBPI Business Process Dashboard you can now:

- See that you have one active instance of the `Orders` process
- Drill down into the `Healthy Instances` selection, and then drill into your process instance. You should be able to see the actual process diagram for this instance and see the associated (related) data instance

You should see a process diagram similar to this:



The `New Order` step has completed and the process instance is currently sitting active in the `Process Order` step.

Injecting More Events

You can now send in further events for this instance and see how your process progresses:

- Send in the Event: `Orders/Processed`

Setting:

```
OrderNumber: Ord1
```

- Back in the HPBPI Business Process Dashboard, refresh your browser page and you should see the process instance has moved on to the `Ship Order` step.

On the process instance page, you can also scroll the page down to see the associated data values - so you can see the `OrderNumber`, `CustomerID`, `IsProcessed`, etc. properties and their values.

- Send in the `Orders/Shipped` event for `OrderNumber Ord1`
Your process instance should now show as completed.
- If you move back to the `Business Process & Resource Summary` page you should now see that your instance is showing as a completed instance.

Note: Within the Business Process Dashboard, it is best to use the navigation bar (“bread crumbs”) across the top of the page to move backwards. If you just use the Web browser’s “back” button the page is shown but not refreshed. By using the bread crumbs to move back, the page is rerun and shows you the latest information

- Now enter a new order as follows:

```
CustomerID    = Cust2
OrderNumber   = Ord2
Value         = 20.11
```

...and move this through the process using the Process Simulator.

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

3 Using BAC

In the last chapter you defined a high-level process that monitors your orders as they move through your business. This enables you to see and measure things, such as:

- Order volumes
- Backlogs at any particular steps
- Time taken for each step
- etc.

Now, what about the IT service dependencies? How do you configure them within HPBPI? Actually..you don't! All the IT service dependencies for your HPBPI process are modeled within Business Availability Center.

Indeed, the integration between HPBPI and Business Availability Center provides more than just the connection of your IT service hierarchy to your business process. With Business Availability Center you also see the current values and volumes moving through your process and its steps, as well being able to produce historical reports on this data.

This chapter looks at how to use HPBPI with Business Availability Center.

Configuring the Connection to BAC

To enable the integration between HPBPI and Business Availability Center, you need to configure the connection from within the HPBPI Administration Console.

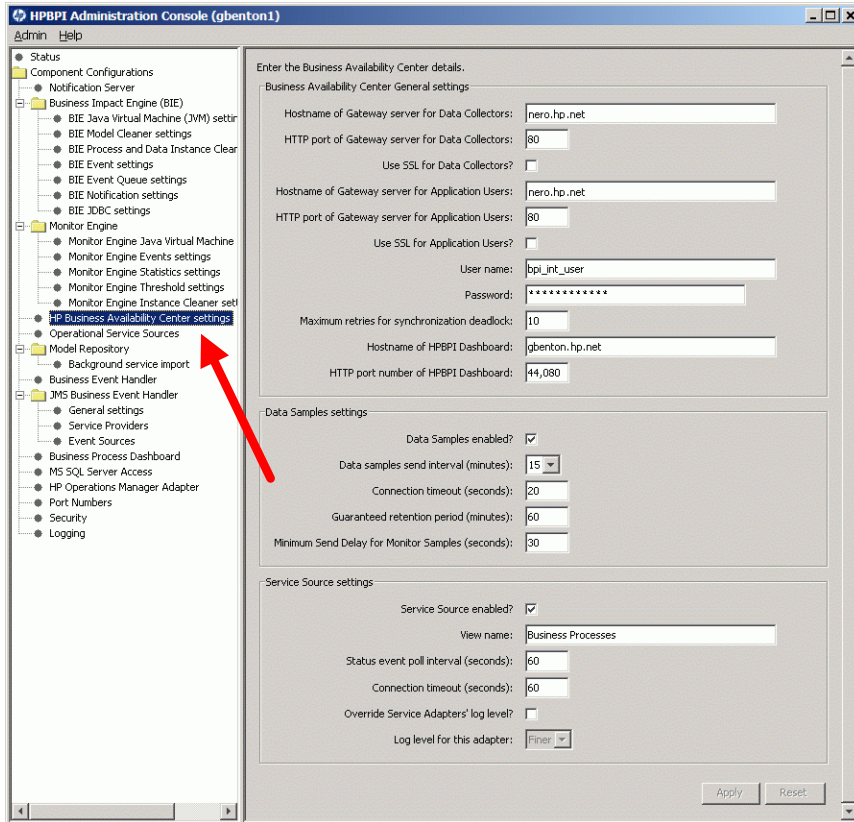
To start up the Administration Console, you select:

Start->Programs->HP->HP Business Process Insight->Administration

Then, within the left-hand menu pane, click on the option:

HP Business Availability Center settings
as shown in [Figure 7](#).

Figure 7 BAC Integration Screen



The dialog that appears in the right-hand pane is fully described in the *HPBPI Administration Guide*, but let's briefly go through and consider some of the key properties.

The dialog consists of three main sections, so let's discuss them section by section...

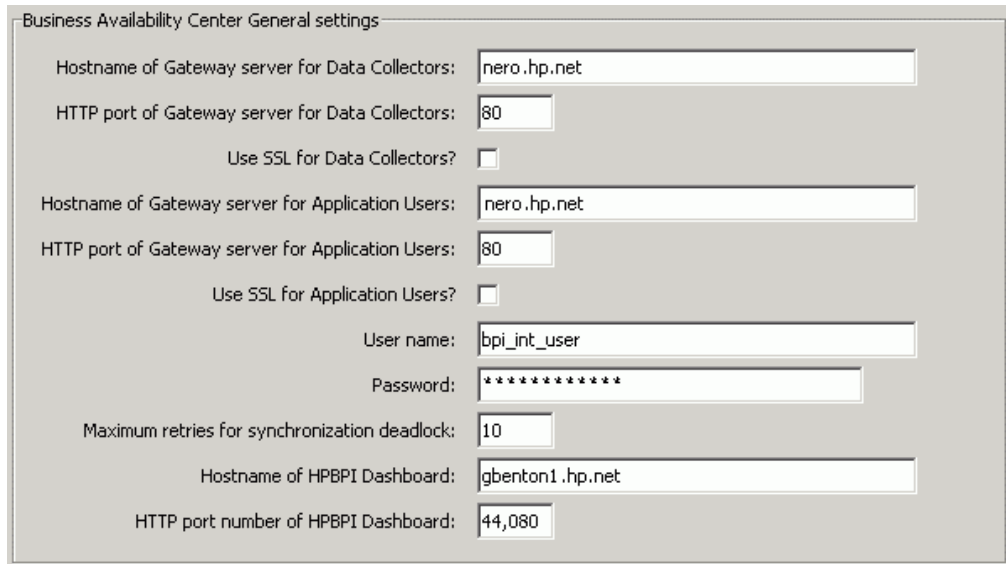
General Settings

The general settings allow you to configure the basic connection details.

The main things you configure here are the host names of the Business Availability Center gateway servers, port numbers, login user name and password.

For example, [Figure 8](#) shows a connection to Business Availability Center running on the host name `nero`, connecting as the Business Availability Center user `bpi_int_user`.

Figure 8 General Settings



The screenshot shows a dialog box titled "Business Availability Center General settings". It contains several input fields and checkboxes for configuring gateway servers and the HPBPI dashboard. The fields are as follows:

Field Label	Value
Hostname of Gateway server for Data Collectors:	nero.hp.net
HTTP port of Gateway server for Data Collectors:	80
Use SSL for Data Collectors?	<input type="checkbox"/>
Hostname of Gateway server for Application Users:	nero.hp.net
HTTP port of Gateway server for Application Users:	80
Use SSL for Application Users?	<input type="checkbox"/>
User name:	bpi_int_user
Password:	*****
Maximum retries for synchronization deadlock:	10
Hostname of HPBPI Dashboard:	gbenton1.hp.net
HTTP port number of HPBPI Dashboard:	44,080

Configuring these general settings does **not** actually enable the integration between HPBPI and Business Availability Center. You are just configuring the Business Availability Center details for when the integration is enabled.

There are two parts to the integration between HPBPI and Business Availability Center:

1. Sending of data samples from HPBPI to Business Availability Center.

This is enabled by filling-in the `Data Samples` settings section of the dialog.

2. Receiving IT operational service status information from Business Availability Center.

This is enabled by filling-in the `Service Source` settings section of the dialog.

You would probably want to enable both of these options to give yourself full HPBPI/Business Availability Center integration.

Data Samples Settings

When you enable data samples, HPBPI starts sending data samples to Business Availability Center. So what data samples are sent?

HPBPI sends a range of data samples including the following:

- The current active count, active value, and throughput, for each deployed process.
- The current active count, active value, and throughput, for each deployed process step.
- The current business health of each deployed process.
- Values and statistics for any HPBPI Monitors that you may have defined for your deployed process(s). Refer to the *HPBPI Integration Training Guide - Defining Business Process Monitors* for details of how to define additional monitors.

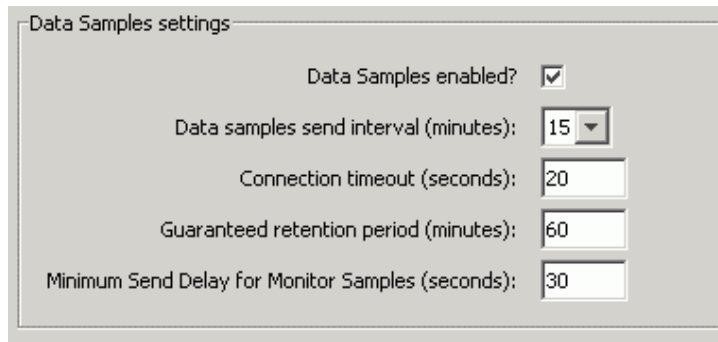
The very first time you enable either the data samples option or the service source option, HPBPI synchronizes all deployed process (and step) definitions with the Business Availability Center uCMDB. For each process and step, configuration items (CIs) are created within the uCMDB. For example, if you have an HPBPI process called `MyProcess` containing a step called `Step1`, then you will have a corresponding CI in the Business Availability Center uCMDB for `MyProcess`, and this will contain a dependent CI called `Step1`.

When HPBPI sends the data samples across to Business Availability Center, these samples are stored against the corresponding CI (process or step) within the uCMDB. You can run reports on this data from within Business Availability Center.

When you enable the sending of data samples, you can specify how often these data samples are sent to Business Availability Center.

For example, [Figure 9](#) configures the sending of data samples to Business Availability Center every 15 minutes.

Figure 9 Data Samples Settings



Setting	Value
Data Samples enabled?	<input checked="" type="checkbox"/>
Data samples send interval (minutes):	15
Connection timeout (seconds):	20
Guaranteed retention period (minutes):	60
Minimum Send Delay for Monitor Samples (seconds):	30

Service Source Settings

When you enable the Service Source option, you are linking your HPBPI process(s) to your Business Availability Center IT hierarchy.

The very first time you enable either the service source option or the data samples option, HPBPI synchronizes all deployed process (and step) definitions with the Business Availability Center uCMDB. For each process and step, corresponding configuration items (CIs) are created within the uCMDB. For example, if you have an HPBPI process called `MyProcess` containing a step called `Step1`, then you will have a corresponding CI in the Business Availability Center uCMDB for `MyProcess`, and this will contain a dependent CI called `Step1`.

Once the processes and steps are defined as CIs within Business Availability Center, you can use Business Availability Center to set up dependencies within the uCMDB between these CIs and other CIs. For example, you could configure the state of a process step CI to be dependent upon the state of a particular database or application system.

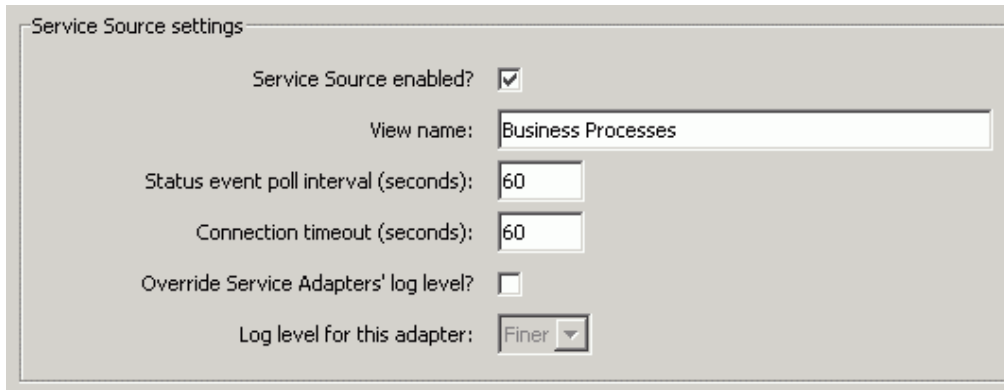
When you enable the service source option within HPBPI you also configure the View name and the Status event poll interval. The view name is the name of a view that must exist within Business Availability Center. HPBPI polls Business Availability Center (as configured by the poll interval) and retrieves the status details for all CIs within the specified view. HPBPI then pulls out the status for each CI and sets this as the status for the corresponding step with HPBPI.

For example, if you have a process called `MyProcess` containing a step called `Step1`, then you will have corresponding CIs in the Business Availability Center uCMDB for `MyProcess` and `Step1`. Thus, when the status of `Step1` goes critical within the uCMDB, HPBPI sets the step called `Step1` within the process `MyProcess` to have a service status of `critical`.

By default there is a pre-defined view within Business Availability Center called `Business Processes`, and this view is configured to include all process and process step CIs. Thus, by default, all HPBPI processes deployed to Business Availability Center appear within the `Business Processes` view.

[Figure 10](#) shows an example of how to enable the service source option, setting the Business Availability Center view name to `Business Processes` and the poll interval to `60` seconds.

Figure 10 Service Source Settings



The image shows a dialog box titled "Service Source settings". It contains the following configuration options:

- Service Source enabled?**
- View name:**
- Status event poll interval (seconds):**
- Connection timeout (seconds):**
- Override Service Adapters' log level?**
- Log level for this adapter:**

Troubleshooting Service Source Configuration

When you configure the Service Source settings within the HPBPI Administration Console, and press apply, your HPBPI processes should start receiving (and showing) any IT service status changes from Business Availability Center - once you have configured IT dependencies within Business Availability Center.

If after configuring all this you find that IT status changes within Business Availability Center are not being correctly reflected within your HPBPI processes, you may need to check the configuration details as specified within your HPBPI Administration Console. If you are still unsure as to what the problem might be, then you can manually issue the API call that HPBPI makes to Business Availability Center, and see first-hand what the problem might be.

Let's look at how you can manually issue the API call to retrieve the IT service status information from Business Availability Center...

To retrieve the IT status information from Business Availability Center, HPBPI makes an HTTP request using the Business Availability Center BAMOpenAPI. This request is issued by the `Service Adapters` component of HPBPI.

If you look at the log file for the HPBPI `Service Adapters` component you can see the actual URL that is used to request the IT status from Business Availability Center.

You look for the line that shows the `BAC URL` that is being used for the `BAC Adaptor`. For example:

```
INFO: Adaptor BAC Adaptor will use BAC URL
http://nero.hp.net:80/topaz/bam/BAMOpenApi?command=getNodes&
customerName=Default%20Client&userName=bpi_int_user&
password=ENCODED-PASSWORD&viewName=Business%20Processes&depth=2&
isEncrypt=y&addRecallAtts=y&layout=hierarchical.
```

(this is all shown on a single line within the log file :-)

This URL is "almost" usable...however...you need to do the following things:

- Remove `isEncrypt=y` parameter.

By removing this parameter you are then able to set the user password using plain text.

- Provide the correct password.

Notice that instead of showing the password within the URL, you see the text ENCODED-PASSWORD. You need to replace the ENCODED-PASSWORD string with the real Business Availability Center user password

You can then start up a Web browser and enter this URL. Make sure you do not include the trailing full-stop (.) from the log file. That is not part of the URL.

If the URL is successful you will receive an XML list of your Business Availability Center service hierarchy for the specified view. If you are not successful you will receive an error message.

For example, [Figure 11](#) shows a successful URL configuration.

Figure 11 Manually Retrieving the IT Status

```

- <nodes user="admin" pass="r00ABXQABlxjwrggBA==" cust="1" appPath="http://nero.hp.net/topaz/" view="Business Processes">
- <node name="Orders" id="84944fa314cc0f82dde512252d4e8422" cmdbId="bfc314d02816cdb4aab99bbe60abbe25">
  <dimension name="Volume" status="-2" id="1050">null</dimension>
  <dimension name="Backlog" status="-2" id="600">null</dimension>
  <dimension name="Business Health" status="-2" id="602">null</dimension>
  <dimension name="Availability" status="20" id="7">null</dimension>
- <children>
- <node name="End" id="5acbf3fe8c9f1faafeba3c2f5ec52ea0" cmdbId="eb404c8118868e162e9891630eea35bb">
  <dimension name="Volume" status="-2" id="1050">null</dimension>
  <dimension name="Backlog" status="-2" id="600">null</dimension>
  <children />
</node>
- <node name="New Order" id="ecf046a1ec2c82707a878208832cc5c2" cmdbId="254e6d743eb703d98ca94d10b30d0476">
  <dimension name="Volume" status="-2" id="1050">null</dimension>
  <dimension name="Backlog" status="-2" id="600">null</dimension>
  <children />
</node>
- <node name="Orders_monitor" id="4bbf8f6f77f246f4251a17493887a6ee" cmdbId="5a6caa6818c1bc20fd0d5ad756645631">
  <dimension name="Volume" status="-2" id="1050">null</dimension>
  <dimension name="Backlog" status="-2" id="600">null</dimension>
  <dimension name="Business Health" status="-2" id="602">null</dimension>
  <children />
</node>
- <node name="Process Order" id="cd0bea11d0c26f79691506314343ff9e" cmdbId="ae00c6773daa4356741289acac355180">

```

A Worked Example

Let's consider the `Orders` process that you defined in [Chapter 2, Process Definition](#).

Your `Orders` process is currently deployed to the HPBPI Business Impact Engine and you have created some process instances using the Process Simulator contributed library.

Let's walk through how to enable the integration with Business Availability Center and then look at what is available to you.

This worked example provides instructions so that you could follow this example through as a lab if you wanted to...

Configure the Integration

As explained in the section [Configuring the Connection to BAC](#) on page 100, you run the HPBPI Administration Console and configure the integration with Business Availability Center.

For this worked example you configure both the Data Sample and Service Source options.

The configuration dialog might look as shown in [BAC Integration Configuration within HPBPI](#) on page 108.

Figure 12 BAC Integration Configuration within HPBPI

Enter the Business Availability Center details.

Business Availability Center General settings

Hostname of Gateway server for Data Collectors:

HTTP port of Gateway server for Data Collectors:

Use SSL for Data Collectors?

Hostname of Gateway server for Application Users:

HTTP port of Gateway server for Application Users:

Use SSL for Application Users?

User name:

Password:

Maximum retries for synchronization deadlock:

Hostname of HPBPI Dashboard:

HTTP port number of HPBPI Dashboard:

Data Samples settings

Data Samples enabled?

Data samples send interval (minutes):

Connection timeout (seconds):

Guaranteed retention period (minutes):

Minimum Send Delay for Monitor Samples (seconds):

Service Source settings

Service Source enabled?

View name:

Status event poll interval (seconds):

Connection timeout (seconds):

Override Service Adapters' log level?

Log level for this adapter:

When you click the `Apply` button to apply this new configuration, because this is the first time you have configured this integration, HPBPI connects to the specified Business Availability Center server and synchronizes any currently deployed HPBPI processes. That is, your `Orders` process is automatically synchronized with Business Availability Center and CIs created within the uCMDB.

Once the integration is configured, if you deploy any new processes, these are automatically synchronized with Business Availability Center at deploy time.

The CIs within BAC

To see the CIs defined within Business Availability Center, you would do the following:

- Logon to Business Availability Center.

For example:

```
http://nero.hp.net:80/topaz
```

where you specify the host name and port number for your BAC server.

- From within Business Availability Center, select:

```
Applications | Dashboard
```

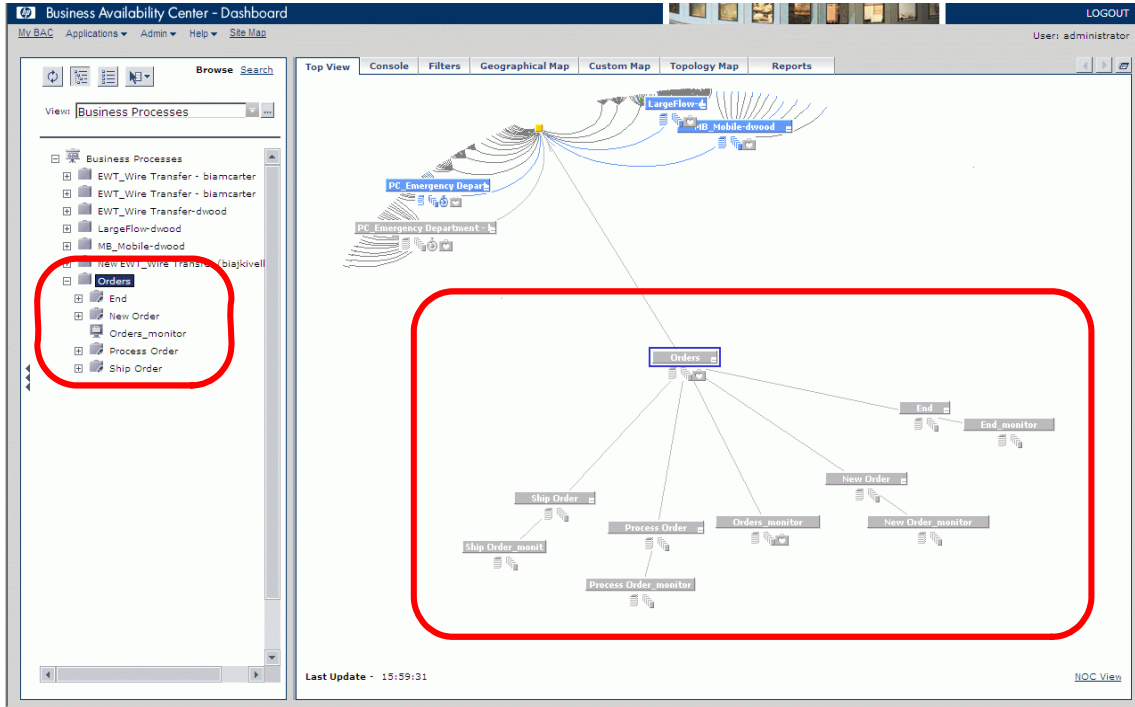
- Select the view `Business Processes` from the pull-down.

This is the default view within Business Availability Center that groups together all HPBPI processes.

- Select the `Top View` tab.

The dashboard should look similar to that shown in [BAC Dashboard - Top View](#) on page 110.

Figure 13 BAC Dashboard - Top View



where:

- You see your `Orders` process, and its steps, in the left-hand menu.
- You see the `Orders` CI and the dependent CIs in the right-hand pane.
- In this example there are other processes deployed as well as the `Orders` process, hence you see these other processes within the `Business Processes` view of the uCMDB.

Notice that HPBPI has created two CIs for each part of your `Orders` process.

Why Two CIs for each Process/Step

For the process you have both an `Orders` CI and an `Orders_monitor` CI, and for each step you have a step CI and a corresponding `step_monitor` CI.

For each process or step, HPBPI defines a CI within the uCMDB so that you can then configure, within the uCMDB, any dependent CI hierarchy. HPBPI also wants to send data samples to the uCMDB for this CI. However, within

the uCMDB, data samples can only be assigned to a “monitor” CI. So for each HPBPI step or process, HPBPI creates both a base CI and a dependent monitor CI.

The CI whose name ends in `_monitor` reflects the data sample values as sent from HPBPI. These values are then inherited by the base CI.

Data Samples

Let's now inject some events into your `Orders` process and see what data is shown in the HPBPI Business Process Dashboard, and what data is reflected in the Business Availability Center dashboard...

Start the Process Simulator Injecting Events

- Run the Process Simulator
- Open the test suite: `sols\orders_testcases.txt`

This simply contains the definitions necessary to inject orders into HPBPI with ever increasing unique order IDs.

- Click on the `Test Suite Runner` tab
- Click on the `Start Suite` button (at the bottom of the screen)

You now have a series of orders being sent through HPBPI.

In the HPBPI Business Process Dashboard

- Start up the Business Process Dashboard
- Drill into the `Orders` process

If you keep refreshing the Web browser you should see the numbers above the process steps changing, indicating that instances are being processed.

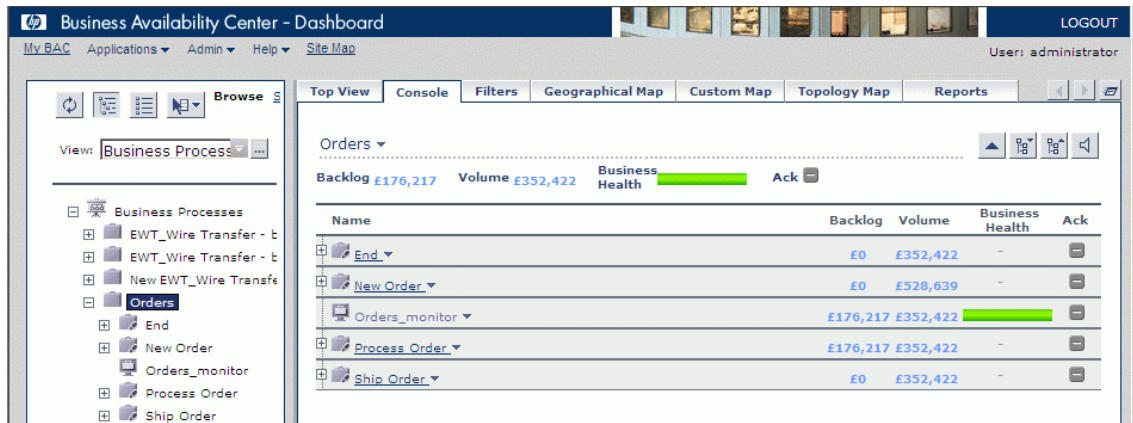
In the BAC Dashboard

- Login to the Business Availability Center Dashboard
- Drill into the Business Process view
- Select the Orders process

In the right-hand pane, you see the overall status of your Orders process showing the Backlog (active process value), Volume (throughput) and Business Health. You also see the Backlog and Volume for each of the steps of the process.

For example, your Business Availability Center Dashboard might look as shown in Figure 14.

Figure 14 BAC Dashboard - Data Samples



where:

- The Backlog is the “active value” for the step or process.
- The Volume is the “value throughput” for the step or process.

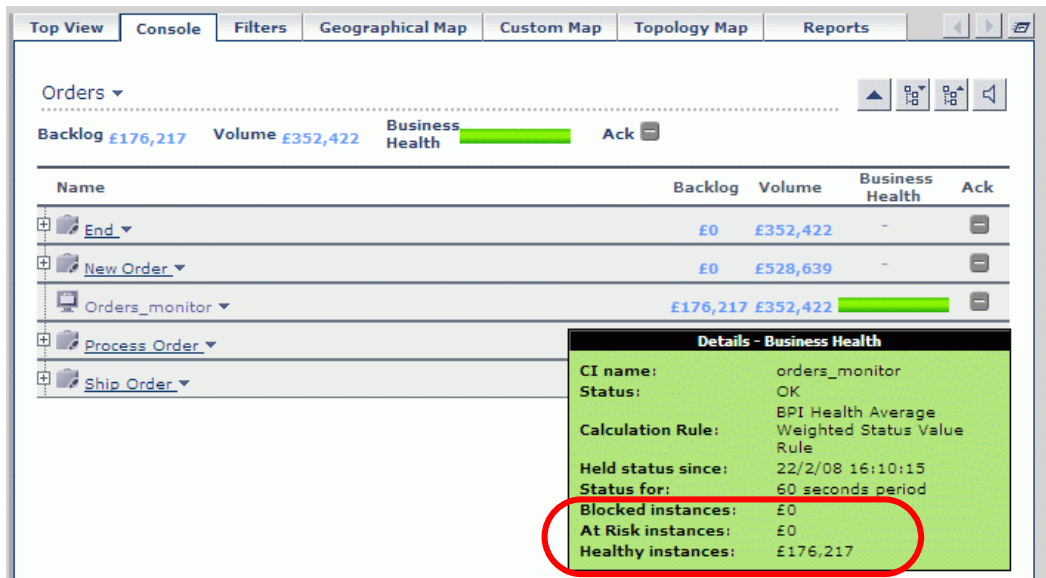
This value is sent from HPBPI and represents the value throughput for the data sample poll interval. So in this example, where the poll interval is set to one minute, the throughput is the value processed for this one minute interval.

- In this example the Business Health bar shows that all current process instances are healthy.

If some instance were blocked or at-risk, then this Business Health bar would provide a visual representation of the relative sizes.

You can also hold the mouse over the Business Health bar and the tooltip shows you the actual values for the healthy, at-risk and blocked instances, as shown in [Figure 15](#).

Figure 15 Business Health Tooltip



Value or Counts?

By default, the Business Availability Center Dashboard is configured to show the various statistics (Backlog, Volume, Business Health) based on the “process value” from your process definition. You can alter this behavior by reconfiguring the KPIs within the Business Availability Center Dashboard.

For example...

- Login to Business Availability Center
- Select:

Admin | Dashboard

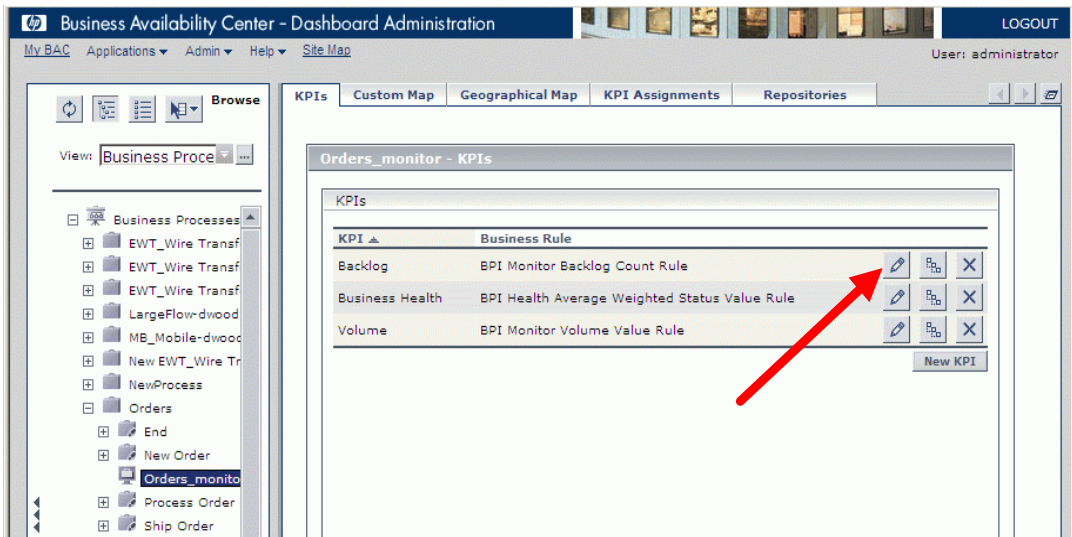
- Within the Business Processes view, drill down to the monitor that you wish to reconfigure.

For example, you could drill down to the monitor `Orders_monitor`.

- You can then configure the KPI that you wish to show within the Dashboard.

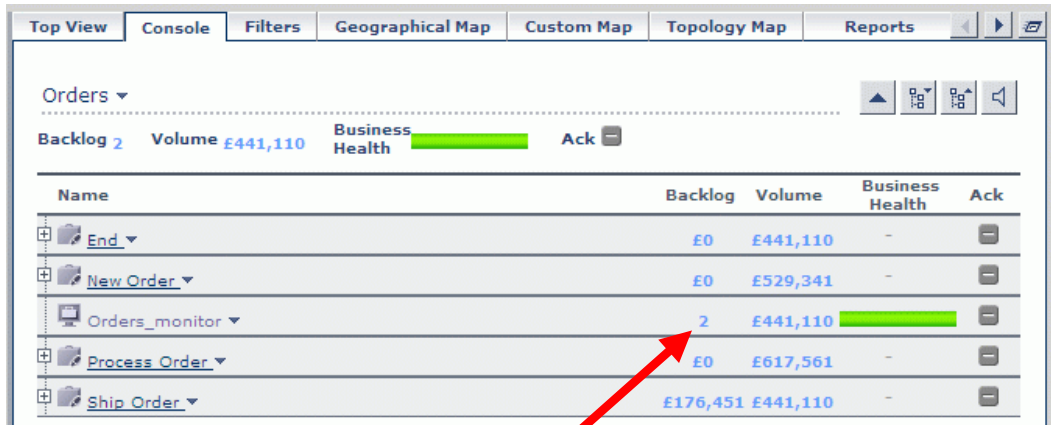
For example, [Figure 16](#) shows that by clicking on the pencil icon, you can select to show the BPI Monitor Backlog Count Rule.

Figure 16 Switching KPIs



Now, when you go back to the Business Availability Center Dashboard application, the backlog for the `Orders` process monitor is displayed as an instance count rather than a dollar value...as shown in [Figure 17](#) on page 115.

Figure 17 Backlog Counts



The screenshot shows a software interface with a table of backlog counts. The table has columns for Name, Backlog, Volume, Business Health, and Ack. The 'Orders_monitor' row is highlighted with a red arrow pointing to its Backlog value of 2.

Name	Backlog	Volume	Business Health	Ack
End	£0	£441,110	-	-
New Order	£0	£529,341	-	-
Orders_monitor	2	£441,110	Green bar	-
Process Order	£0	£617,561	-	-
Ship Order	£176,451	£441,110	-	-

So be aware that for all the CIs in the uCMDB, there are a range of KPIs available to you.

Business Health When There Are No Active Instances

HPBPI sends data sample values to Business Availability Center as often as is configured within the HPBPI Administration Console. In this example, these data samples are being sent every minute.

If there are no active instances of the process (or any of its steps) at the exact moment that HPBPI sends a data sample, then the Business Health within Business Availability Center will show as “unknown”.

For example, your Business Availability Center dashboard might look as shown in [Figure 18](#) on page 116.

Figure 18 No Current Business Health Shown

The screenshot shows a software interface with a menu bar at the top containing 'Top View', 'Console', 'Filters', 'Geographical Map', 'Custom Map', 'Topology Map', and 'Reports'. Below the menu bar, there is a section for 'Orders' with a dropdown arrow. To the right of this section are three icons: a triangle pointing up, a square with a downward arrow, and a square with an upward arrow. Below this is a summary row with the following data: 'Backlog £0', 'Volume £53,331', 'Business Health' (with a small circular icon), and 'Ack' (with a minus sign icon). The main part of the interface is a table with the following columns: 'Name', 'Backlog', 'Volume', 'Business Health', and 'Ack'. The table contains five rows of data:

Name	Backlog	Volume	Business Health	Ack
End	£0	£53,331	-	-
New Order	£0	£53,331	-	-
Orders_monitor	£0	£53,331	-	-
Process Order	£0	£53,331	-	-
Ship Order	£0	£53,331	-	-

A red arrow points to the 'Business Health' column for the 'Orders_monitor' row, which contains a small circular icon.

Service Source

In this example, the integration with Business Availability Center also enables the Service source settings. This allows the linking of your HPBPI process(s) to your Business Availability Center IT hierarchy.

Let's see how it works...

Adding Dependencies

Within the Business Availability Center administration tool you can configure dependencies for the CIs of your process steps. For example, you might configure that your `Process_Order` CI is dependent upon the status of a CI that represents a database or application system, or whatever. The way you link up your CI dependencies is up to you.

In this worked example, the `Ship_Order` step CI has been set up to be dependent on a Business Availability Center *Availability* KPI.

Service Status Change in BAC

Suppose the status of the availability KPI for the `Ship_Order` CI changes to *Critical*. Within the Business Availability Center dashboard you would see this status change as shown in [Figure 19](#).

Figure 19 BAC Dashboard - Service Critical

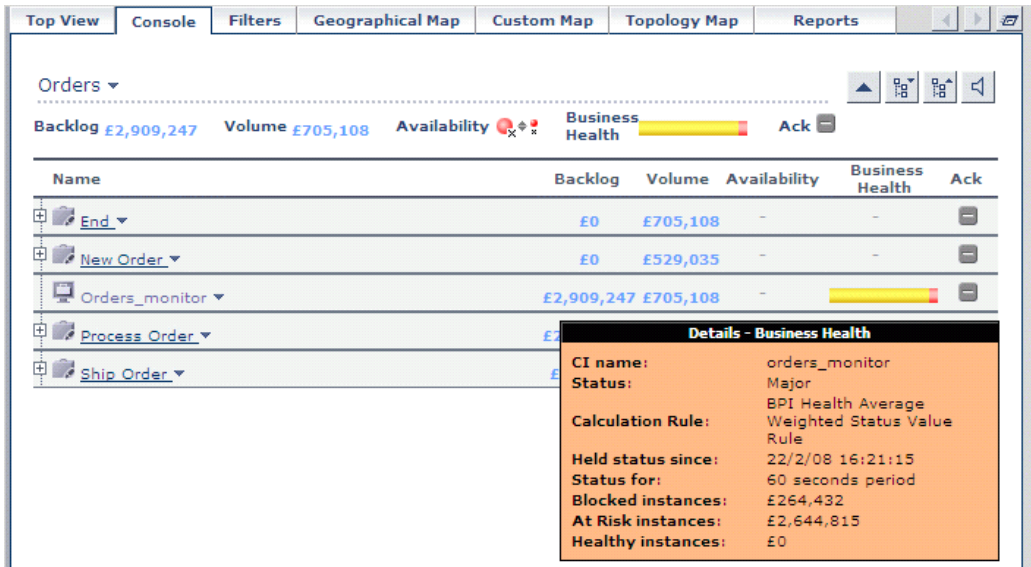


Notice that in [Figure 19](#) the `Ship_Order` CI shows that the *Availability* is indeed Critical.

Notice also that the Business Health shows a “yellow and red bar” that indicates that you now have some process instances at-risk (yellow) and some process instances blocked (red). The relative sizes of the colors in this bar indicate the relative percentage of at-risk and blocked. If there were some process instances that were healthy, then the Business Health bar would also contain a section of green - to indicate the relative amount of healthy instances.

You can hover the mouse over this Business Health bar to see a tooltip that displays the actual sizes of the healthy, at-risk and blocked instances...as shown in Figure 20.

Figure 20 Business Health Tooltip

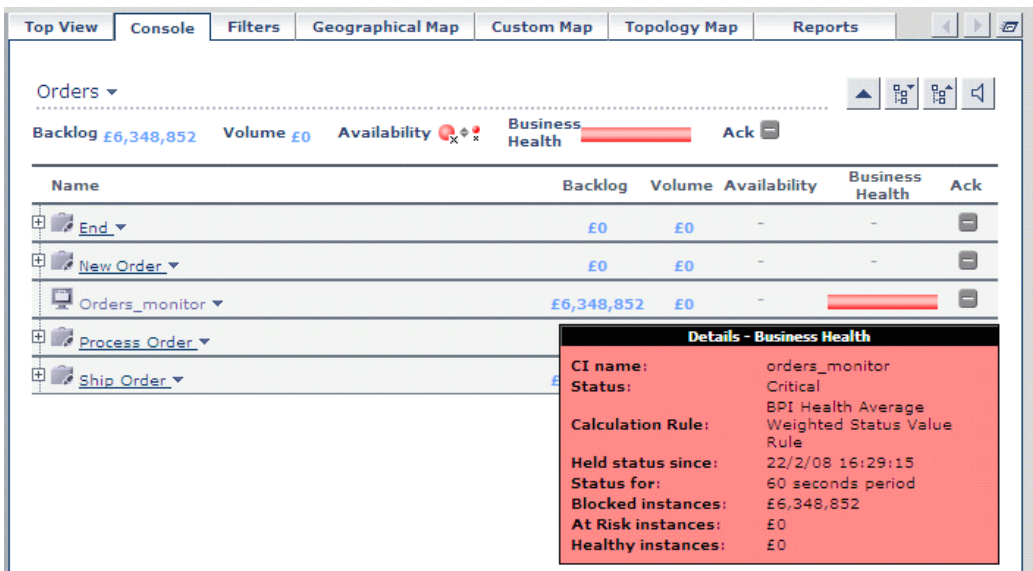


Notice that Figure 20 shows the actual values for the amount of healthy, at-risk and blocked instances.

You might also be wondering why the Business Health tooltip shows as a status of Major (with an orange background) when there are blocked process instances? The reason is that the default rule assigned within Business Availability Center for showing this tooltip uses a calculation based on the percentage values of healthy, at-risk and blocked instances. This calculation determines what the resultant overall status for the tooltip should be set to. In Figure 20 the calculation has decided that as there are only a small percentage of blocked instances, that the overall status for this tooltip is Major.

If you have mainly blocked instances, then the Business Health tooltip shows as a red background, showing a state of critical - for example, see [Figure 21](#).

Figure 21 Business Health Tooltip - Critical



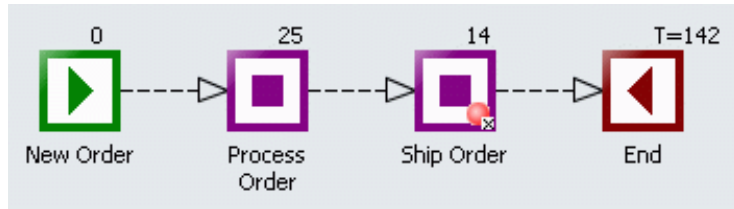
By default, the Business Availability Center Dashboard is configured to show the size of the healthy, at-risk and blocked process details as values. You can alter this behavior by reconfiguring the KPIs within the Business Availability Center Dashboard administration tool.

Service Status Change in HPBPI

When a process step CI changes its status within Business Availability Center, this status is reflected back within HPBPI. Remember that HPBPI is polling (by default) the Business Processes view within Business Availability Center, and this enables it to retrieve the status change for all the process and step CIs.

In this example, when the `Ship_Order` CI changes to Critical within Business Availability Center, this change is reflected back in HPBPI, and you can see this in the HPBPI Business Process Dashboard. The `Orders` process might look as shown in [Figure 22](#) on page 120.

Figure 22 HPBPI Dashboard - Ship Order Step Critical



If you click on the `Ship Order` step, it brings up the details for this process step. You can then see that the service that has caused this step to go critical is the BAC service called `Orders - Ship Order` - as shown in [Figure 23](#).

Figure 23 HPBPI Dashboard - Step and Service Details

HPBPI Dashboard

Home > Orders > Ship Order

Process Step and Service Summary

Process Step		Service Summary		
Name	Ship Order	Service Type	Name	State
Overall Service Status	x	BAC	Orders - Ship Order	x
Average Step Completion Time (H:M:S)	0 : 01 : 42.22			

You can then click on this `Orders - Ship Order` link to drill into the Service Health page, as shown in [Figure 24](#) on page 121.

Figure 24 HPBPI Dashboard - Service Health

HPBPI Dashboard

Service Health

Home > Orders - Ship Order

Orders - Ship Order from adapter "BAC Adaptor"

Current status of Orders - Ship Order [View in BAC Console](#)

Last Change	Current status	Description
25-Feb-2008 17:12:42 GMT		Current status of service Orders - Ship Order is critical.

Usage of Orders - Ship Order

Process Name	Step Name
Orders	Ship Order

The Service Health screen contains the [View in BAC Console](#) link. Clicking on this link takes you across to the Business Availability Center dashboard, showing you the details for the Ship Order CI.

Synchronization Matters

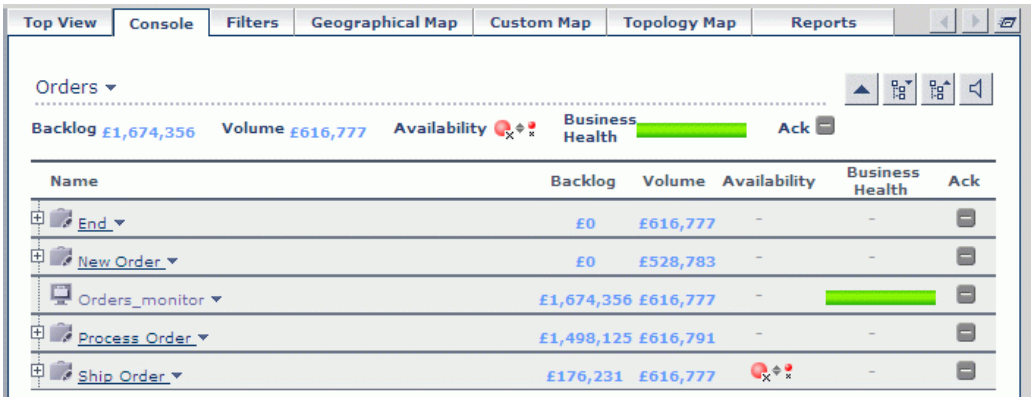
As you have seen in [Figure 19](#) on page 117, the Business Availability Center dashboard displays the Business Health of a process. The data that determines the coloring of how this Business Health bar is displayed, is sent from HPBPI. This data is sent from HPBPI according to how you configure the Data samples settings within the HPBPI Administration Console - see [Figure 7](#) on page 100.

In this worked example, the Data samples settings are configured to be every one minute.

However, the Business Availability Center dashboard also shows the status of your CIs within the uCMDB, and these statuses are displayed as soon as they change within the uCMDB.

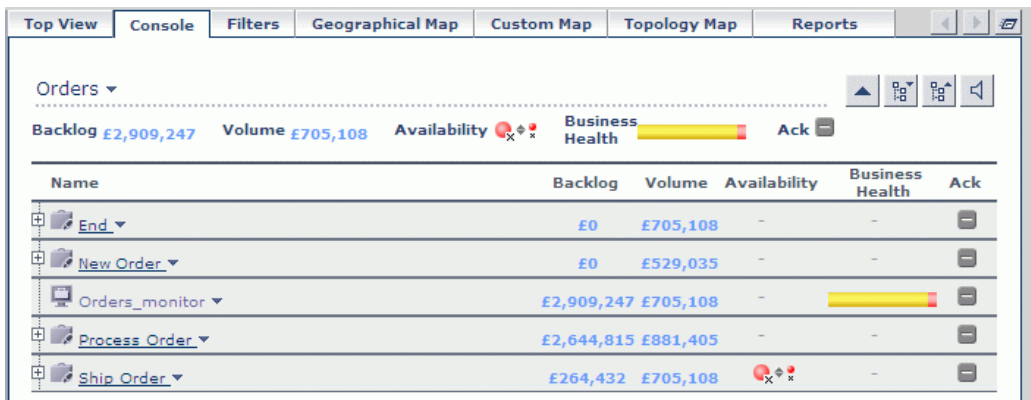
So it is quite possible that when you view you process statistics within the Business Availability Center dashboard that you might see a CI status change, but not see the Business Health bar change to reflect this until some time afterwards. For example, your Business Availability Center dashboard might look as shown in [Figure 25](#).

Figure 25 Business Health and CI State - Out of Sync



In [Figure 25](#), you see that the status of the Ship_Order CI has changed to Critical within the uCMDB, however, the Business Health bar is showing that no process instances are affected. Once HPBPI and Business Availability Center have synchronized, this screen will be updated and the true status reflected. For example, the status might be updated as shown in [Figure 26](#).

Figure 26 Business Health and CI State - Synchronized



Summary

The integration between HPBPI and Business Availability Center provides the following key features:

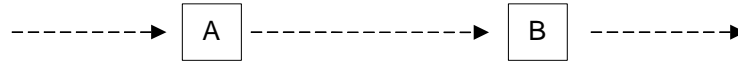
- Your process(s) and steps are automatically modeled within the uCMDB as CIs.
- All the IT dependencies for your HPBPI process steps are modeled within the uCMDB.
- HPBPI receives status changes from the uCMDB and is able to determine the business impact of these status changes.
- The Business Availability Center dashboard is able to display the latest process statistics, including backlogs, volumes, business impact/health.
- The Business Availability Center dashboard is able to maintain these process statistics over time, providing you with the ability to produce historical and user-defined reports.

4 Advanced Process Definition

This chapter covers some more advanced topics of process definition.

Actual Step Ordering

Suppose you have defined the following process segment:



It is important to realize that there is no guarantee that the steps activate in the order as shown in the diagram.

The process diagram is what you believe is going to happen. At run time, the HPBPI Business Impact Engine actually tells you what has happened and the order in which the steps actually occurred.

The connecting arcs are just there for you to indicate the ordering that you expect to occur. The actual step progression is dictated by your step progression rules and the order in which the work actually occurs in your underlying application systems.

Trapping Steps Out Of Sequence

If you do wish to dictate the ordering of some, or all, of the steps, and trap any times when this ordering is not obeyed...you can.

When you define a connecting arc in the HPBPI Modeler, you can right-click on the arc and select that you wish to **Check Sequence**.

This changes the arc to be a solid line (rather than a dashed line).

For example:



This tells the HPBPI Business Impact Engine (at run time) to trap any situation where step B is started before step A. If step B does start before step A then HPBPI issues a **Process Out Of Sequence** notification. (The destination of this notification can be configured within the HPBPI Notification component.)

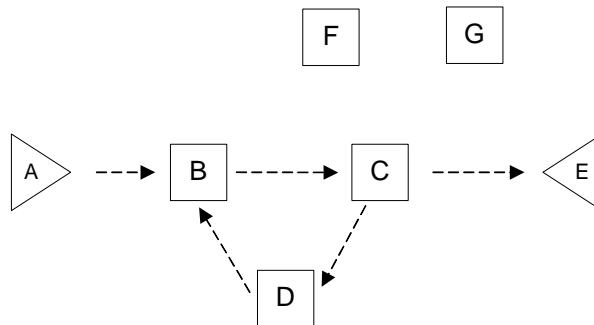


Note that the out of sequence condition is only tested when step B starts. That is, if step B completes (without having first started) before step A has started or completed, then this does not cause an out of sequence trap to be caught.

Steps With No Arcs

There is nothing to stop you designing a business process where you have one or more steps that are not connected by arcs to any other steps.

For example:



where:

- The process should normally progress along the path: A, B, C, D, B, C, E
- But at any stage in this progression the process could switch out to either step F or G

Rather than trying to draw lines from every step to F, and then every step to G, just show them as “stand alone” steps within the process.

A stand alone step is started/completed according to its start/complete progression rules - just like any other step.

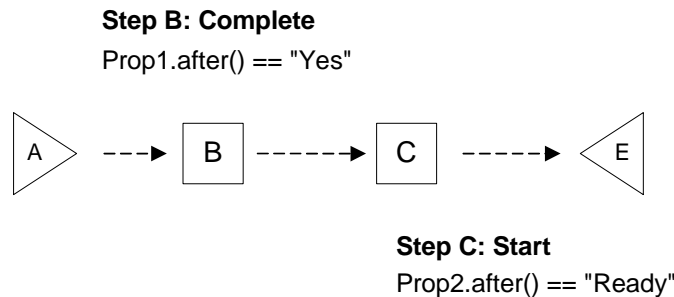
Active Processes With No Active Step(s)

It can happen that you deploy a process and then, when monitoring it with the HPBPI Business Process Dashboard, you notice that you have an active process instance however it does not show as being active within any of the steps of that process.

For example, you might have one active process instance, but when you list the number of process instances at each step you see all steps showing that they have no active instances.

This can seem very confusing at first sight. It can also make you feel that perhaps HPBPI has got it wrong. No...it is all to do with your step progression rules.

Let's consider this example process:



- When the process enters step B the properties are in the state:
Prop1 = "No"
Prop2 = null
- When the event comes in to set Prop1 to "Yes" you then have the following property values:

Prop1 = "Yes"
Prop2 = null

Which means that the step B completion rule is satisfied - so the process instance moves out of step B.

However, Prop2 did not change to "Ready" so the process instance does not move in to step C.

Thus the HPBPI Business Impact Engine has an active process instance - and this shows in the HPBPI Business Process Dashboard allowing you to see its existence - however the HPBPI Business Process Dashboard is not able to show you where the process is at the moment.

This situation normally occurs because you have “gaps” in your process. That is, you are not representing all the stages/transitions of the process. There is nothing wrong about this. HPBPI quite happily monitors your process and reports what it can. It just might surprise you when/if it happens.

If you do have this situation where you have active process instances that are not showing up in your step instance list, then it might mean that you need to model more of your business process. Or it might mean that you have made an error in one or more of your progression rules. Of-course, it might be exactly what you want :-)

Activity Steps Starting Processes

You would normally expect to design a process with a start step and then have that start step actually start the process instance. And, by default, that is the normal behavior - where a process instance is only created when the start step’s start condition is satisfied.

However, you can also design your process to allow a normal activity step(s) to start your process instance.

If you open up the properties dialog for an activity step you see two check boxes available to you:

The image shows a screenshot of a software dialog box titled "Step Properties - Assigned". The dialog has three tabs: "Step", "Progression", and "Services". The "Step" tab is selected. It contains the following fields:

- Name: Assigned
- Description: (empty)
- Type of Step: Activity (dropdown menu)
- Two checkboxes at the bottom, both of which are circled in red:
 - OK to start the process if not already started
 - Log a warning if the process has not already started

where:

- The “OK to start the process...” check box

This applies in the situation where the start condition for this step has been satisfied, however, there is no current process instance.

This tells the HPBPI Business Impact Engine that if this start condition is satisfied, and there is no current process instance, then go ahead and instantiate one.

In other words, this step is able to start the process instance if the process instance has not already been started.

- The “Log a Warning...” check box

You can check this only if you have checked the “OK to Start...” check box.

This tells the HPBPI Business Impact Engine that you would like to log a warning message whenever this step does actually instantiate a new process instance.

The log message is logged to the HPBPI Business Impact Engine’s log file, which can be viewed using the HPBPI Administration console.

The message is logged as a warning. For example:

```
WARNING: Entering a process instance at step "Assigned" when  
         the instance did not already exist.
```

Junction Steps

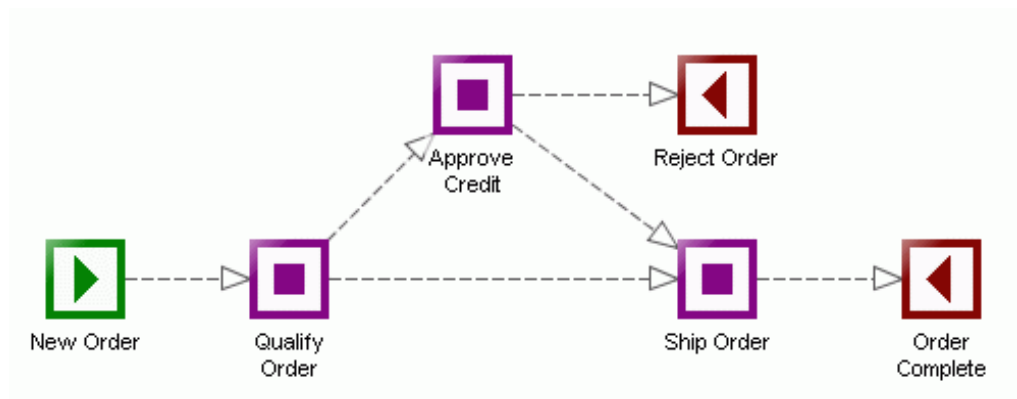
A junction step can be placed anywhere within a process diagram. You can have more than one junction step within a process diagram. You may choose to have no junction steps within your process. Junction steps are optional.

So why does HPBPI have Junction steps?

You may use a Junction step to represent a step within the process that you are choosing not to monitor. You may use a Junction step to represent a decision that takes place within the underlying process. Some people may find that using Junction steps simply helps them layout their process diagram. Junction steps are purely there to aid with the visual layout of your process diagram.

For example, suppose you were modeling an order system. You might draw the process as shown here in [Figure 27](#):

Figure 27 Order Process Diagram



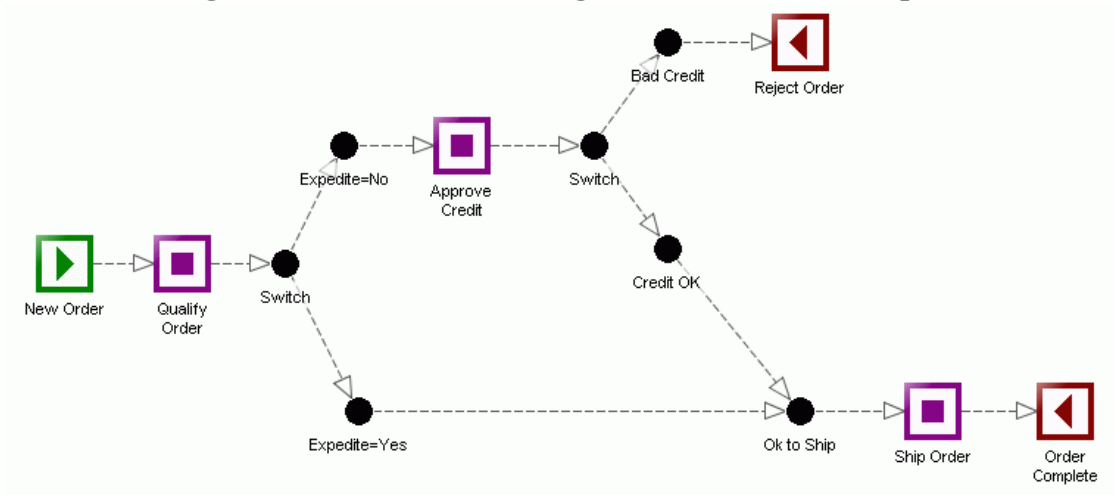
The above process shows that when an order comes in, you first qualify the order to see if the customer is a special customer or not. In the case of a special customer's order, you choose to expedite this order as there is no need to approve their credit. You can go straight ahead and ship the order. For all non-special customers, you first approve their credit, and if they are approved, you ship the order, else you reject the order.

Some people might feel that the diagram in [Figure 27](#) does not fully show these decision points. That is, how do you know, simply from looking at the diagram, that the Approve Credit and the Ship Order steps are not

carried out in parallel? Also, the diagram does not obviously show why an order might take the path through the `Approve Credit` step as opposed to the `Ship Order` step.

You can use Junction steps to add this additional information. For example, you could redraw the order process as shown in [Figure 28](#):

Figure 28 Order Process Diagram with Junction Steps



Now you can see that after the order is qualified, there is a decision which sends the order one way or the other. If the order is marked to be expedited, then the order goes through one path, else it goes through the other path. The Junction steps are there purely to help show the structure and decision points within your process diagram.

A Junction step has no progression rules and has no dependencies. It is simply a step that you can add to the process diagram to help you represent the layout of the process.

The name of a Junction step does not need to be unique within the process diagram. You can have duplicate Junction step names.

The Model Repository

When you are running the HPBPI Modeler you are connected to the Model Repository.

Repository Startup

The Model Repository is started/stopped by the HPBPI Administrator console.

At startup, the Model Repository performs a number of consistency checks that are worth understanding about.

When the Model Repository starts up it tries to check the deployed state of all its definitions - Processes, Data, Events. To do this it needs to connect and talk to the HPBPI Business Impact Engine. Hence it is always a good idea to start the HPBPI Business Impact Engine component before starting the Model Repository.

The Model Repository connects to the HPBPI Business Impact Engine and finds out whether the definitions that it thinks are deployed are indeed still deployed to the Engine. If the HPBPI Business Impact Engine has (for example) undeployed a process, the Model Repository is able to flag that this process is no longer deployed.

Thus, if you restart the Model Repository and then re-run the HPBPI Modeler you see a clean indication of which process/data/etc. definitions are actually deployed to the HPBPI Business Impact Engine.

However, be careful because if the Model Repository is started when the HPBPI Business Impact Engine is not running then, when you run the HPBPI Modeler, you see **all** your processes and data definitions appear to be **undeployed!**

Importing And Exporting Definitions

The HPBPI Modeler allows you to import and export definitions.

The most obvious example is where you have defined a process - and associated data, events, etc. - on a test machine. You now want to export this all to a file, take this file to your production server, import it and deploy it.

Exporting

To export a definition from within the HPBPI Modeler:

- Click on the definition you wish to export (in the Navigator pane)
- Then select:

File->Export Definition...

- Then specify the file to which you want the export saved

This exports the selected definition **and all related definitions**.

The definition(s) are saved to a zip file.

For example, if you export a process, this exports the process definition and any associated data definitions, event definitions, etc. It exports everything required by that process.

You can also export the entire Model Repository in one go, using the HPBPI Repository Explorer - see [Exporting Definitions](#) on page 138 for more details.

Importing

To import definitions into the HPBPI Modeler:

- Select:

File->Import Definitions...

- Then choose the input file to import.

You can select either a previously exported (.zip) file, or a file containing a Business Process Execution Language (.bpel) file.

If you have selected a ZIP file:

- You are then presented with a list of all the objects to be imported. If there are any clashes with current definitions you have the ability to decide whether to overwrite/rename/ignore these definitions.

If you have selected a BPEL file:

- You are presented with a range of import options which allow you to affect how the import is performed.

- When you are happy with your import options...click the `Import` button

Any definitions that you import, come into the HPBPI Modeler as being **undeployed**. It is then up to you to deploy this definition to the HPBPI Business Impact Engine.

Repository Explorer

As well as the HPBPI Modeler, there is an additional tool that allows you to view the Model Repository, it is the HPBPI Repository Explorer.

The HPBPI Repository Explorer is a set of JSP pages that display the contents of the Model Repository within a Web browser window. You cannot create or modify definitions from within the HPBPI Repository Explorer, but there are a number of features that are quite useful.

Running the Repository Explorer

There are two ways to run the HPBPI Repository Explorer:

1. Standalone
2. From within the Modeler

Standalone

To run the HPBPI Repository Explorer standalone, run a Web browser and go to the URL:

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

where:

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

You are prompted for a user name and password. These are the same user name and passwords as required for the HPBPI Modeler. The default user name and password details are:

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


From Within the Modeler

You can launch the HPBPI Repository Explorer from within the HPBPI Modeler.

You can either select:

Window->Repository Explorer

from the menu item, or simply click on the HPBPI Repository Explorer icon:



Choosing either of these options launches a Web browser with the correct URL specified. You still need to logon as a valid repository user.

View Details

The HPBPI Repository Explorer lets you view your definitions. The details are displayed for each object all on a single screen. This means that you can see a process's definition, including all the progression rules, on a single screen. The same is true for viewing data definitions. You can view all the event subscriptions together on one screen. This can be very useful when developing or troubleshooting a definition.

The HPBPI Repository Explorer also lets you view the details of **superseded** definitions. Select a definition, and then click the `History` tab (in the right-hand frame). For example, this allows you to see the process diagram of a superseded process, or the data properties of a superseded data definition. You can also walk through the links from a process definition to its corresponding data definition.

Recycle Bin

When you delete definitions within the HPBPI Modeler these definitions are **not physically deleted**. They are simply marked as deleted within the Repository. Once marked as deleted, the definition are no longer visible from within the HPBPI Modeler.

To see the definitions currently marked as deleted, you use the HPBPI Repository Explorer and look in the `Recycled` bin.

Cleanup Recycled

You can individually remove items from the Recycled bin, or you can ask to remove all items from the Recycled bin.

To remove all items from the Recycled bin, you:

- Click on the `File` icon within the `Navigator` pane of the `HPBPI Repository Explorer`
- Then select the `Cleanup Recycled` option

Restore

If you have accidentally deleted a definition from within the `HPBPI Modeler`, you can go into the `HPBPI Repository Explorer` and restore that definition from the Recycled bin.

Exporting Definitions

The `HPBPI Repository Explorer` lets you export a definition hierarchy, or the entire `Repository` (excluding the `Recycled bin`).

Export

To export a single definition hierarchy:

- Select the definition (in the `Navigator` frame) you wish to export
- Click on the `File` icon, within the `Navigator` frame.
- Select `Export`

In the right-hand frame you see the list of definitions that are to be exported.

- Click the `Download` button in the right-hand frame

Export All

To export the entire contents of the HPBPI Model Repository - excluding items in the Recycled bin:

- Click on the `File` icon, within the Navigator frame.
- Select `Export All`

In the right-hand frame you see the list of definitions that are to be exported.

- Click the `Download` button in the right-hand frame

Deployment

When you select a definition to deploy, the HPBPI Modeler deploys the selected definition and any dependencies. This is why, when you deploy a process, the HPBPI Modeler deploys all the parts necessary to enable that process to be deployed and run.

There is nothing stopping you deploying an event definition. In which case, only that event definition is deployed. You can deploy just a data definition - and the HPBPI Modeler deploys that data definition and any as yet undeployed event definitions that this data definition is subscribed to.

But don't forget that the HPBPI Modeler only deploys definitions if they have been changed since the last time they were deployed. Thus, the first time you deploy a process, the HPBPI Modeler deploys everything. You then make a change to the process definition (maybe add a new step). You then re-deploy this process definition...and the HPBPI Modeler only deploys the process definition. This is because the related data and event definitions have not changed and thus there is no need for them to be redeployed. There is nothing wrong with this behavior - it just surprises some people when they first notice it.

Superseded Processes

When you redeploy a process, the HPBPI Business Impact Engine wants to delete the previously active version of the process. And if there are no process instances (active or completed) for this old process definition...the HPBPI Business Impact Engine deletes it.

However, what happens when the old version of the process still has some active (or completed) process instances? The HPBPI Business Impact Engine cannot simply delete them - the active process instances are still running and still receiving data events. So, the HPBPI Business Impact Engine marks this old process definition with a status of "Deleted" - which actually means "Pending Delete". What this means is that, as and when this process definition no longer has any active or completed process instances, it is then deleted from the HPBPI Business Impact Engine.

These "Deleted" processes are called **Superseded** processes.

Suppose your process name is: `ProcA`.

- At any one time there can be many versions of `ProcA` deployed to the HPBPI Business Impact Engine
- Only one of these (the most recently deployed) is marked as the “Active” process definition. All the rest are superseded (marked as “Deleted”)
- When an event comes in to start a new instance of `ProcA`, the HPBPI Business Impact Engine always starts the version marked as “Active” - the most recently deployed version
- All instances of superseded process definitions run through to completion as normal

As events come in they are routed to the correct process instance and run according to that process definition. Redeploying a process definition does **not** affect running instances.

- Over time, as process instances complete, these may be removed from the HPBPI Business Impact Engine by the “Instance Cleaner” (refer to the *HPBPI System Administration Guide* for details about the instance cleaner).

Once all instances of a superseded process definition have been removed, the actual process definition is then removed.

Intervention Client

If you have superseded definitions (processes/data) and you would like to remove them from the HPBPI Business Impact Engine...you can.

You use the **HPBPI Intervention Client**.

The HPBPI Intervention Client is a set of JSP pages (and java classes) that allow you to view the deployed definitions and instances within the HPBPI Business Impact Engine...and update/delete them.

▶ You need to be **very careful** when using the HPBPI Intervention Client. If you use it incorrectly you can seriously mess up the HPBPI Business Impact Engine and HPBPI Modeler.

To run the HPBPI Intervention Client, you point a Web browser at the URL:

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

where:

- *hostname* is the hostname of your servlet engine. This is usually the same as the HPBPI server hostname
- *44080* is the default port of the servlet engine

By default these JSP pages are password protected. The default login details are:

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

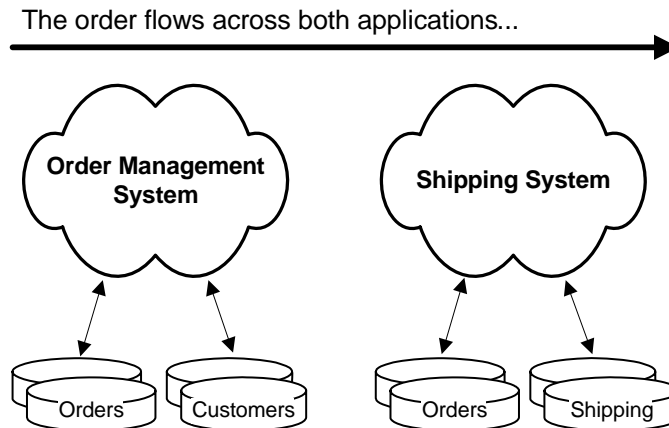
The HPBPI Intervention Client is a useful utility for process developers. It allows you to deploy your processes and data as often as you need to get them right, and then you can use the HPBPI Intervention Client to remove all the superseded definitions and thus clean up your HPBPI Business Impact Engine. Very useful!

▶ If you are wanting to clean up your development system using the HPBPI Intervention Client, then it is recommended that you use the HPBPI Intervention Client to remove the definitions and all instances from the running HPBPI Business Impact Engine **and then re-deploy** the process(s) that you wish to use afresh.

Multiple Unique IDs

When defining a data definition, you normally set up the definition to have a single unique ID attribute. This means that each data instance is identified by a single unique ID for the life of the data instance. For example, each order is known by its `OrderID`.

But how do you handle the situation where you are monitoring (for example) an order as it moves across disparate application systems? For example:



While the order moves around the Order Management System, it may be uniquely identified by an `OrderID` attribute. But when the order moves across to the Shipping System, the order may well be known by not just a different attribute name, but also a different unique ID value.

For example, an order within the Order Management System known by the key `OrderID = 123`, might become known within the Shipping System by the key `ShippingOrderNum = ABC-987`. While this order is in the Order Management System you receive events identified by the `OrderID (123)`. When this order moves around the Shipping System, you receive events identified by the `ShippingOrderNum (ABC-987)`. You need a way for the HPBPI Business Impact Engine to know that `OrderID=123` and `ShippingOrderNum=ABC-987` are indeed the same data instance.

To be able to track this order as it moves across these two applications, you need to:

- Create your data definition to have two unique ID attributes.

Create two attributes, `OrderID` and `ShippingOrderNum`, and mark each one as being unique. This says that a particular data instance can be identified by either a unique `OrderID` value or a unique `ShippingOrderNum` value.

- Get the applications people to provide a mechanism for sending an event into the HPBPI Business Impact Engine that maps between these two key values.

When an order moves from the Order Management System across into the Shipping System, you need to receive an event containing both key attributes. This allows the HPBPI Business Impact Engine to identify that the data instance identified by `OrderID=123`, now has the ID `ShippingOrderNum=ABC-987`.

Setting up this mechanism to send an event when an order moves across to the Shipping System may take some time and effort, but you need some way to send an event that contains both the `OrderID` value and its new `ShippingOrderNum` value.

So, for example, your event sequence might become:

```
New Order           (OrderID=123, ...)  
Update Order       (OrderID=123, ...)  
X-Application Send (OrderID=123, ShippingOrderNum=ABC-987)  
Update Shipping    (ShippingOrderNum=ABC-987, ...)  
Shipping Complete  (ShippingOrderNum=ABC-987, ...)
```

where:

- The first two events (`New Order` and `Update Order`) are uniquely identified by the `OrderID` attribute, and the value is `123`.
- When the order is sent from the Order Management System to the Shipping System, the event `X-Application Send` provides the mapping that `OrderID 123` is now known as `ShippingOrderNum ABC-987`.
- The last two events provide information about the order as it moves through the Shipping System. Each event is identified by the `ShippingOrderNum` attribute, whose value is `ABC-987`.

Handling Out of Sequence Events

When working with events from across multiple application systems, the ordering of the events becomes crucial.

For example, with the event sequence:

```
New Order           (OrderID=123, ...)  
Update Order       (OrderID=123, ...)  
X-Application Send (OrderID=123, ShippingOrderNum=ABC-987)  
Update Shipping    (ShippingOrderNum=ABC-987, ...)  
Shipping Complete  (ShippingOrderNum=ABC-987, ...)
```

It is the X-Application Send event that tells the HPBPI Business Impact Engine that, from now on, events for ShippingOrderNum=ABC-987 are to be linked to the same data instance identified by OrderID=123.

However, what happens if the events arrive in this order:

```
New Order           (OrderID=123, ...)  
Update Order       (OrderID=123, ...)  
Update Shipping    (ShippingOrderNum=ABC-987, ...)  
X-Application Send (OrderID=123, ShippingOrderNum=ABC-987)  
Shipping Complete  (ShippingOrderNum=ABC-987, ...)
```

When the New Order event arrives, a new data instance is created for OrderID=123. When the Update Order event arrives, this updates the same data instance where OrderID=123. When the Update Shipping event arrives the HPBPI Business Impact Engine tries to locate an existing data instance where ShippingOrderNum=ABC-987. It does not find one...so it creates a new data instance setting ShippingOrderNum=ABC-987. You now have two data instances - one with OrderID=123 and another with ShippingOrderNum=ABC-987. Unfortunately, this is not what you wanted! Because the X-Application Send event had not made it through to the HPBPI Business Impact Engine in time, the HPBPI Business Impact Engine had no way to know that ShippingOrderNum=ABC-987 should have been tied to the existing data instance where OrderID=123.

But you can solve this issue by correctly setting your event subscriptions. See [Retry If Data Instance Doesn't Exist](#) on page 146

Retry If Data Instance Doesn't Exist

In the case where you are monitoring across more than one application system, and therefore the item being monitored is known by more than one unique ID, you need to configure your process model **not** to automatically create new data instances all the time.

For the example where the events are:

```
New Order           (OrderID=123, ...)
Update Order        (OrderID=123, ...)
X-Application Send  (OrderID=123, ShippingOrderNum=ABC-987)
Update Shipping     (ShippingOrderNum=ABC-987, ...)
Shipping Complete   (ShippingOrderNum=ABC-987, ...)
```

Within the HPBPI Modeler, you set up the data definition subscriptions for each of these events.

For the New Order, Update Order and X-Application Send events, you can accept the default setting:

An instance can already exist. Create if necessary.

However, for the remaining Update Shipping and Shipping Complete events, you set the creation option to be:

An instance must first be created by another event. Retry this event later on.

The subscription page looks something like this:

Figure 29 Retry the Event

1 Associate with a specific instance

Event is for a specific instance of this data definition.

Unique Property of this Data Definition: == Property of Event:

2 In order to handle the event, should the data instance already exist?

where:

1. The event subscription is for a specific instance of the data definition.
2. You select that the event **retries** if the specific data instance does not yet exist.

Setting the option to `Retry` means that the HPBPI Business Impact Engine marks this event to be re-submitted, and hands this event back to the HPBPI Event Handler. The event is periodically re-sent over a configurable time period. The idea is that during this retry time the missing event (that sets up the data instance) arrives.

So by marking an event subscription to “Retry” if the data instance does not yet exist, allows you to receive the events out of order but the HPBPI Business Impact Engine processes them in the correct order.

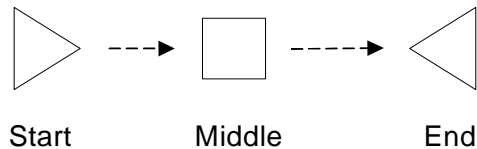
Lab - The Process Seems To Stop Working

This lab looks at a typical developers problem where you have a process that works, you make some minor changes, and suddenly your process appears to have stopped working.

The Process

Let's create a simple process:

- Create a process called: `Simple`
- Draw it out to look like this



- Create a data definition called: `Simple/Data`
- Define the data properties as follows:

Name	Type	Constraints	Unique
<code>Key_Field</code>	<code>String</code>	10	Yes
<code>Value_Field</code>	<code>Integer</code>		

- Relate `Simple/Data` to `Simple`
- Set the `Identity` to be `Key_Field`
- Set the `Process Value` to be `Value_Field`

- Set the progression rules as follows:
 - Start
 - Complete on first assignment:
 - Property: Key_Field
 - Middle
 - Start and complete on transitions:
 - Start Transition:
 - Property: Value_Field
 - From:
 - To: 15
 - Complete Transition:
 - Property: Value_Field
 - From: 15
 - To: 66
 - End
 - Complete on transition:
 - Property: Value_Field
 - From:
 - To: 66

These progression rules simply mean that when a key comes in, it moves into the `Middle` step when its value field is set to `15`. It then moves into the `End` step when the value field is set to `66`.

Nice and simple :-)

- Define an event called: `Simple/Event`
- Define the properties for this event to be both the `Key_Field` and `Value_Field` as defined in the `Simple/Data` definition
- Configure `Simple/Data` to subscribe to this `Simple/Event` as follows:
 - It is a specific subscription matching on `Key_Field`
 - The subscription assigns:


```
this.Value_Field = event.Value_Field;
this.Key_Field   = event.Key_Field;
```

- Check your `ToDo` lists to make sure that nothing is missing
- Deploy `Simple`

You should see it deploy:

- `Simple/Event`
- `Simple/Data`
- `Simple`

Testing The Process

- Run the HPBPI Business Process Dashboard

You should be able to see that your process `Simple` is deployed and currently has no active instances.
- Using the Process Simulator contributed utility and inject the following event:

```
Event name:    Simple/Event
Key_Field:    123
Value_Field:  15
```

- Now in the HPBPI Business Process Dashboard you should hopefully be able to see this new process instance and see that it is indeed sitting in the `Middle` step.

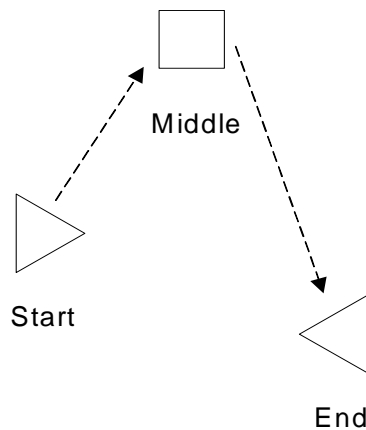
Wonderful! Your process is working.

Altering The Process Definition

Let's suppose that you are happy with your process and how it all works, but you are about to give a demonstration to your manager and you would like to make some alterations to the process's appearance.

So...let's go back into the HPBPI Modeler and make some changes...

- In the HPBPI Modeler, edit the `Simple` process definition and simply move the steps about on the screen so that they are now positioned as follows:



That's right. All you do is to alter the positions of the steps on the page.

The important thing is that you are updating the process definition.

- Save your changes

Notice that this causes the icon next to `Simple` (in the Navigator pane) to change color.

- Now select `Simple` in the Navigator pane...and deploy the process

Notice that it only redeploys the process definition, as that is the only thing that has changed.

Testing The New Process

Ok, you have redeployed the process. You know that it worked before so you are all ready to do your demonstration for your manager. Good luck!

Let's do the demonstration...

- Run the HPBPI Business Process Dashboard

If you refresh the process page, you should now see that `Simple` is listed in both the active processes table and the superseded processes table.

And your new active version of `Simple` has no instances as yet.

- Using the Process Simulator, you inject the following event:

Event name: `Simple/Event`

Key_Field: `123`

Value_Field: `15`

- You then look in the HPBPI Business Process Dashboard and you don't see a new instance for `Simple!!!`

At this point you look very embarrassed and you explain to your manager that "It worked this morning...honest!"

The Explanation

The problem is very simple.

When you first deployed your process and tested it, you used a key value of 123. This created a data instance in the HPBPI Business Impact Engine for the unique key of 123. This was then connected to a process instance of `Simple`.

When you redeployed `Simple`, you only redeployed the process definition.

Now when you inject an event with `Key_Field` of 123, the HPBPI Business Impact Engine sees that it is for the data instance whose unique field (`Key_Field`) is 123. And guess what? You already have one of these.

Updates to this data instance 123 affect both the superseded and the newly deployed process `Simple`, however it depends on the progression rules for the new `Simple` as to what behavior you see. In this case, sending in the event to set the `Value_Field` to 15 does not cause a new process to be instantiated.

If instead you had injected an event with a new unique key value (for example: `Key_Field = 456`) you would have seen a new instance of your newly deployed process `Simple` and everything would have been fine!

There are a couple of things to realize here:

- Firstly, when injecting your own “made up” events you need to be **very careful** about your unique key fields

If you are injecting events that require new unique values, make sure that they are unique, otherwise you may not see new process instances as you expect.

- If you really do wish to ensure that you can reuse previously tested unique keys you have an option available to you:

If you change the process definition, make sure you also change the data definition (just changing the description should be enough).

This causes the data definition to be redeployed as well as the process definition. This means that events received instantiate new data instances and you are able to see new process instances created as you expect.

However, be aware that these events with previously used unique key values also affect the superseded process instances as well. So it is not necessarily a good thing to do on a production system.

You might want to consider using the **HPBPI Intervention Client** contributed utility to cleanup superseded process and data definitions - and then redeploying the process. Then when you give your demonstration everything works correctly because there is no “old data” left hanging around.

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

5 Advanced Progression Rules

When configuring your step progression rules you have various *styles* available to you. There are a collection of “typical” (or easy) styles and then you have the `Advanced conditions` style.

The set of typical/easy styles are fine for developing your processes and the use of them is certainly encouraged. However, depending on the complexity of your process, you may need to configure advanced progression rules for some, or all, of your steps.

You typically need to write advanced progression rules when your start (or complete) condition for a step relies on testing the values of more than one data property.

This chapter looks at how you write advanced progression rules. This chapter also looks at the issue of how to ensure your progression rules are robust enough to cope if/when your business data events arrive out-of-sequence.

The Language

When writing progression rules you are writing the evaluation(s) that you want the HPBPI Business Impact Engine to perform for your step each time a data event comes in that affects your process's related data definition.

So, as you write each progression rule, you must always be asking yourself:

“When the event comes in, does this rule evaluate to true or false.”

Your start condition, if it evaluates to true, means that the step is marked as started. Your complete condition, if it evaluates to true, means that the step is marked as completed.

The Grammar

The details of the actual grammar for the rules language are all specified in the *HPBPI Reference Guide*.

When writing progression rules you are writing the “test condition” part of an “if” statement. The elements that you can test are the state of the data definition instance itself and/or the state of the properties of this related data instance.

To write these tests there are a set of available methods that you can use. The trick is knowing what each method does and how to use it...

Data Definition Level Methods

You can test the state of the related data instance.

The important methods available to you are:

- **.created()**

This evaluates to boolean `true` when the data instance has just been created.

This might typically be used as the entry condition for a start step.

For example:

Have a start step whose entry condition says:

```
    this.data.created()
```

This starts the step when an event comes in that creates an instance of the related data definition.

- **.terminated()**

This evaluates to boolean `true` when the data instance has just been terminated.

This can occur if the event subscription has been configured to terminate the data instance on receipt of the event. (There is a check box on the event subscription dialog.)

Data Property Level Methods

You have various methods available to test the values of the properties of the data instance.

The important methods available are:

- **.changed()**

This method evaluates to boolean `true` if the value of the data property has changed with this event.

For example:

```
this.data.Call_Status.changed()
```

- **.before()**

This method is for when a data event has come in and changed the value of the data property. By using `.before()` you can test the old value of the property.

For example:

```
this.data.Call_Status.before() == "Assigned"
```

Note that `.before()` only evaluates if the data property actually changed on receipt of this event. If this event did not change `Call_Status`, then this expression evaluates to `false`. Indeed, if the event did not change the property then any expression containing the use of `.before()` on this property evaluates to `false`...no matter what you are comparing it to.

So:

```
this.data.Call_Status.before() == "Assigned"
```

is the same as:

```
this.data.Call_Status.changed()  
&&  
this.data.Call_Status.before() == "Assigned"
```

- **.after()**

This method is for when a data event has come in and changed the value of the data property. By using `.after()` you can test the new value of the property.

Note that `.after()` only evaluates if the data property actually changed on receipt of this event. If the event did not change the property then any expression containing the use of `.after()` on this property evaluates to false.

So:

```
this.data.Call_Priority.after() > 3
```

is the same as:

```
this.data.Call_Priority.changed()  
&&  
this.data.Call_Priority.after() > 3
```

- **.contains()**

This method allows you to test whether the data property contains a specified string.

This only works on string properties.

For example:

```
this.data.property1.after().contains("AbC")
```

This evaluates to true if `property1` has changed and the after value (the new value) is a string that contains the characters `AbC` (case sensitive).

```
this.data.property1.contains("AbC")
```

This evaluates to true if `property1` is a string that contains the characters `AbC`.

- **.starts()**

This method allows you to test whether the data property starts with a specified string.

This only works on string properties.

- **.ends()**

This method allows you to test whether the data property ends with a specified string.

This only works on string properties.

- **.in()**

This method allows you to test whether the data property is set to any one of a list of values.

This method works on string/non-string properties.

For example:

```
this.data.property1.after().in(1,2,3)
```

This evaluates to true if the numeric property `property1` has changed value and is now set to either numeric 1 or 2 or 3.

```
this.data.property2.after().in("E103", "E203")
```

This evaluates to true if the string property `property2` has changed value and is now set to any of the strings `E103` or `E203`.

Comments Within Progression Rules

The logic within advanced progression rules can sometimes get quite complex. To help you document what the progression rule is doing, you can place comments within the progression rule.

There are two types of comments:

1. `/* */`

These allow you to mark the beginning and end of a comment. These comments can extend over multiple lines.

2. `//`

This allows you to mark the start of a comment. This comment ends when a new line is encountered.

If you place one of these comments at the end of a progression rule, there must be a carriage-return after the end of the comment. If you forget to do this, the HPBPI Modeler shows a pop-up dialog to remind you.

For example, you could create a progression rule as follows:

```
// This is my progression rule
/*
 * This tests whether a claim status starts with
 * the letter D
 */
this.data.Claim_State.starts("D") // A comment
&&
( this.data.Claim_Letter_Status.after() == "Drafted"
// another comment
&&
  this.data.Claim_Letter_Type.after() == "Dispute"
/* Some
  more
  comments */
)
// A comment on the last line
```

where:

- The actual progression rule code is shown here in bold type-face.
- The comments are there purely for your own documentation - they do not affect the execution of the progression rule.
- The comment on the last line is followed by a carriage return.

Single Start/Complete Condition

Sometimes you want a step to simply signal that something has happened.

For example, you might have a step called: “Purchase Order Received”, which simply completes the moment you receive the purchase order from the customer.

You could set the step progression rules as follows:

Start Condition:

```
this.data.POReceived.after() == true
```

Complete Condition:

```
this.data.POReceived.after() == true
```

This means that the step both starts and completes the moment the `POReceived` property becomes `true`.

To save you having to specify the same rule for both the start and the completion conditions, if a step has only an entry condition, then when that condition is satisfied the step is said to start **and** complete at the same time. So too, if a step only has a complete condition, then when that condition is satisfied the step is said to both start and complete at the same time.

Steps Re-Starting/Completing Too Often

The way step-start and step-complete progression rules work is easy to understand:

A step is started each time its start condition evaluates to true.

A step is completed each time its complete condition evaluates to true.

Sometimes you deploy a process and observe that one or more of your steps seem to be started (or completed) more than once. Now this might be perfectly expected behavior, however, this can be confusing if you expected the step to start (or complete) only once.

Clearly, if a step shows as starting (or completing) more than once then it's start (or complete) condition was satisfied more than once. If you only expected the step to start (or complete) the once then you have a problem with the logic in your progression rules.

A common error that can cause this behavior is where you have forgotten to use the `.after()` or `.before()` methods of a data property. Let's consider an example:

You have a step where the start condition is when the property `MyProp` transitions to be not null. However, you write the progression rule as follows:

```
this.data.MyProp != null
```

As each event comes in for this data instance, whilst `MyProp` is still `null` the step does not start.

The first time a non-null value is assigned to `MyProp`, the above progression rule evaluates to `true` and the step starts. So far so good!

However, for **every** event thereafter that affects this data instance, the above progression rule evaluates to `true`! That is, this rule is evaluated and the Business Impact Engine says that `MyProp` is indeed still not equal to `null`.

So the progression rule evaluates to true the first time `MyProp` transitions to non-null, and for every event thereafter - assuming that `MyProp` stays set to a non-null value.

The correct progression rule is:

```
this.data.MyProp.after() != null
```

This only evaluates to true on events where `MyProp` has actually been changed and the new value is not null.

So be careful. Remember that **all** step progression rules are evaluated for each event that affects the process's related data instance.

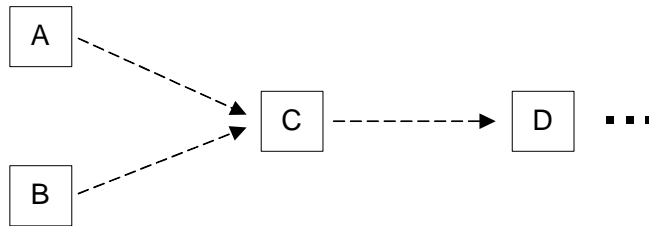
Complex Start Rules

Configuring a progression rule where a step does not start until two or more previous steps have completed, typically requires the use of advanced progression rules...and you need to give these careful consideration.

Example 1

Suppose your process looks as follows:

sets **ValidPONumber** to be true



sets **VendorID** to be non-null

where:

- Step A completes when `ValidPONumber` is set to true
- Step B completes when `VendorID` becomes a non-null value

You want Step C to start only when **both** `ValidPONumber` has transitioned to true and `VendorID` has transitioned to be non-null.

Most people would start by writing the following progression rule to start Step C:

```
this.data.ValidPONumber.after() == true
&&
this.data.VendorID.after() != null
```

and then wonder why Step C never starts :-)

Indeed, the above progression *might* work...but only if the one event caused both the `ValidPONumber` and the `VendorID` to transition.

Remember that the above progression rule is actually saying:

```
(   this.data.ValidPONumber.changed()
  && this.data.ValidPONumber.after() == true
)
&&
(   this.data.VendorID.changed()
  && this.data.VendorID.after() != null
)
```

So this only evaluates to true if both properties have been changed by the one event.

Instead, you need to write a progression rule that caters for the situation where the event has either updated the `VendorID` or the `ValidPONumber`. The progression rule should be written as follows:

```
(   this.data.ValidPONumber.changed()
  || this.data.VendorID.changed()
)
&&
(   this.data.ValidPONumber == true
  && this.data.VendorID      != null
)
```

where this tests:

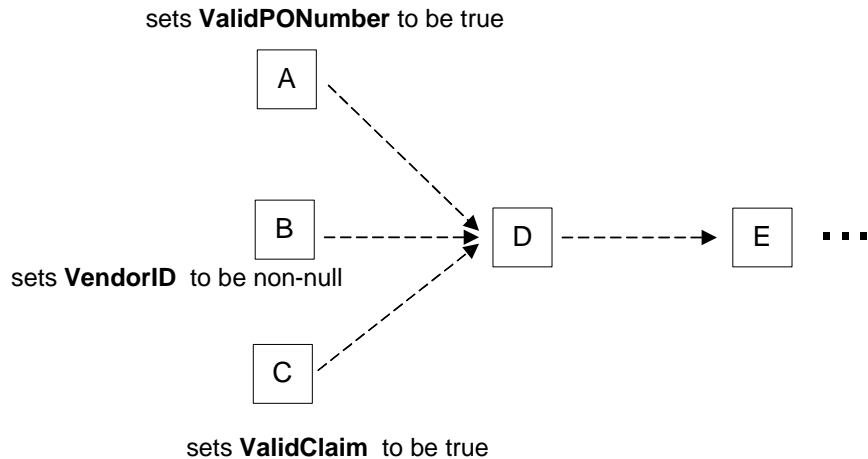
- That at least one (or both) of the attributes `ValidPONumber` and `VendorID` have just been changed by the event.
- And the attributes have the set of values that you are testing for:

```
   this.data.ValidPONumber == true
  && this.data.VendorID     != null
```

- This progression rule also works in the situation where both `VendorID` and `ValidPONumber` are set within the one event.

Example 2

Suppose you have the following process definition:



You need to write the start progression rule for `Step D` where you only enter `Step D` once `Steps A, B and C` have all completed.

The progression rule is as follows:

```
(
  | this.data.ValidPONumber.changed()
  | this.data.VendorID.changed()
  | this.data.ValidClaim.changed()
)
&&
(
  this.data.VendorID      != null
  && this.data.ValidPONumber == true
  && this.data.ValidClaim  == true
)
```

where you test that one or more of the attributes has been changed by this event, and if so, that the values for the three attributes are all set to the values you require.

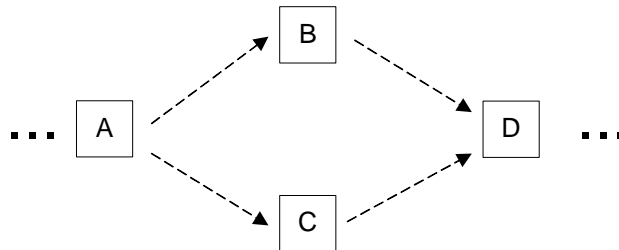
Out of Sequence Business Data Events

Let's look at how to write progression rules that cope when business data events arrive into HPBPI out of sequence.

Consider the following example:

- The process diagram is as shown in [Figure 30](#).

Figure 30 A Sample Process



- The value of a single data attribute, `MyStatus`, determines which step the process instance is in at any one time.

When `MyStatus` is set to `A`, the process instance is in the step `A`. When `MyStatus` is set to `B`, the process instance is in step `B`, etc.

Typical Progression Rules

There are typically two ways that people use to set up the progression rules for the process as shown in [Figure 30](#) on page 168. Both mechanisms use the simplified style of progression `Start` and `complete` on transitions and they are as follows:

1. Active step determined by current value of `MyStatus`.
2. Active step determined by data transition.

Let's consider these mechanisms in more details...

Active Step Determined by Current Value

This is where you say that when `MyStatus` is set to `X` you have entered step `X`, and when `MyStatus` changes from `X` you have left step `X`.

For example, the progression rules for Step B would be as follows:

```
Start transition:
  Property: MyStatus
  From:
  To:      "B"
```

```
Complete transition:
  Property: MyStatus
  From:    "B"
  To:
```

...and so on for all the other steps.

Active Step Determined by Data Transition

This is where you say that when the value of `MyStatus` transitions from A to B you have entered step B, and when the value of `MyStatus` transitions from B to D you have left step B.

For example, the progression rules for Step B would be as follows:

```
Start transition:
  Property: MyStatus
  From:    "A"
  To:      "B"
```

```
Complete transition:
  Property: MyStatus
  From:    "B"
  To:      "D"
```

...and similarly for all the other steps.

Problems If Events Arrive Out of Sequence

Both progression rule mechanisms 1. and 2. described above, work fine if the business data events come into HPBPI in the correct order. For example, if the events come in as:

```
Event 1: MyStatus=A, BPI_GeneratedDate=10:01
Event 2: MyStatus=B, BPI_GeneratedDate=10:02
Event 3: MyStatus=D, BPI_GeneratedDate=10:03
```

then the process works correctly. The process diagram shows the correct start and complete times for each step and everything is ok.



Notice that the business data events are being sent to HPBPI with the `BPI_GeneratedDate` attribute set. This attribute is used to reflect the actual date/time that the business data event occurred in the real world. This is explained further in the *HPBPI Integration Training Guide - Business Events*.

However, if the events arrive at HPBPI out of sequence, then you will have problems.

For example, if the events were to arrive at HPBPI in the following order:

```
Event 1: MyStatus=A, BPI_GeneratedDate=10:01
Event 2: MyStatus=D, BPI_GeneratedDate=10:03
Event 3: MyStatus=B, BPI_GeneratedDate=10:02
```

your process instance would not correctly represent what actually happened in the real world.

As for precisely how the process instance ends up depends on which progression rule mechanism you are using. But the kind of problems you may see include:

- The process instance completed but some steps remain active - when all steps should show as completed.
- The process instance completed but some steps have not even been started - when all steps should show as started and completed.
- The step durations not being correct, causing the process time line diagram to be incorrect.

So how can you cope with out of sequence business data events?

Well Defined Start and Complete Conditions

To be able set up progression rules that cope when the business data events arrive out of sequence you require that, for each step, you have a well-defined start condition and a well-defined complete condition.

In the example process, as shown in [Figure 30](#) on page 168, the `MyStatus` attribute has the possible values A, B, C or D. So these status values enable you to configure well-defined start conditions for each step. That is, when `MyStatus` becomes B, step B has started. But what about the complete condition?

Data Transitions

You might think that you can configure a well-defined complete condition based on the actual data transition. For example, step B completes when `MyStatus` transitions from the value B to the value D. However, data transitions do not behave as you expect, or require, when the business data events arrive out of sequence. Let's walk through an example...

Suppose your progression rules for step B (in the process shown in [Figure 30](#) on page 168) are as follows:

```
Start transition:
Property: MyStatus
From:      "A"
To:        "B"
```

```
Complete transition:
Property: MyStatus
From:      "B"
To:        "D"
```

Now suppose the events arrive in the following order:

```
Event 1: MyStatus=A, BPI_GeneratedDate=10:01
Event 2: MyStatus=D, BPI_GeneratedDate=10:03
Event 3: MyStatus=B, BPI_GeneratedDate=10:02
```

When the first event arrives the value of `MyStatus` is set to A. When the next event arrives, `MyStatus` is set to the value D which is a transition from A to D. When the last event arrives, `MyStatus` is set to the value B - which is a transition from D to B.

So in this situation, step B **never** starts or completes!

Unfortunately, HPBPI does not maintain a time-sequenced set of values for an instance of a data definition. That is, HPBPI only keeps the most recent value for a data attribute. Thus, when event 3 comes in setting `MyStatus` to B, HPBPI is not able to see that the previously received event (2) setting `MyStatus` to D actually happened (in the real world) after the setting of `MyStatus` to B.

Interpreting the Complete Condition

You might consider setting the complete condition for a step to be the same as the start condition of the following step. This can work if business data events arrive out of sequence however, it does not cope if a process step can be followed by multiple steps. Let's explain...

Single Thread Process

Consider the process segment as shown in [Figure 31](#).

Figure 31 Single Thread Process



Suppose you configure the progression rules for step B as follows:

```
Start transition:  
Property: MyStatus  
From:  
To:      "B"
```

```
Complete transition:  
Property: MyStatus  
From:  
To:      "D"
```

Notice that the complete condition shown here for step B is the same as the start condition for step D.

Configuring the progression rules as above means that even if the business data events arrive out of sequence, HPBPI is able to cope with this and the process instance is recorded correctly.

For example, suppose the events arrive in the following order:

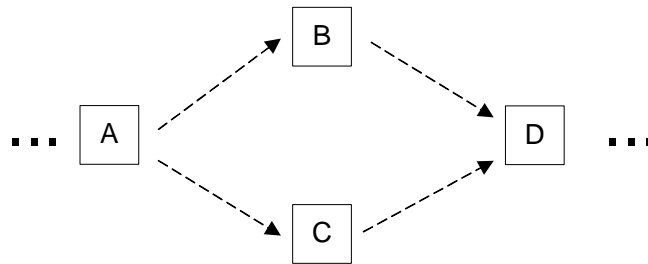
```
Event 1: MyStatus=A, BPI_GeneratedDate=10:01
Event 2: MyStatus=D, BPI_GeneratedDate=10:03
Event 3: MyStatus=B, BPI_GeneratedDate=10:02
```

When the first event arrives the value of MyStatus is set to A. When the next event arrives, MyStatus is set to the value D - this causes step B to be completed and the complete time is set to 10:03. When the last event arrives, MyStatus is set to the value B - and this causes step B to be started and the start time is set to 10:02. Because the start time is before the complete time, HPBPI shows this as a completed step and the time line diagram reflects the correct start and complete times...and the duration for this step is correctly recorded.

Multi Threaded Process

Now consider the process segment as shown in [Figure 32](#).

Figure 32 Multi Threaded Process



When you configure the complete progression rules for each step to be the same as the start condition of the following step, you encounter problems.

For example, the complete progression rules for steps B and C both end up being the same. That is...

Step B is as follows:

```
Start transition:
Property: MyStatus
From:
To:      "B"
```

```
Complete transition:
Property: MyStatus
From:
To:      "D"
```

Step C is as follows:

```
Start transition:
  Property: MyStatus
  From:
  To:      "C"
```

```
Complete transition:
  Property: MyStatus
  From:
  To:      "D"
```

Notice that the complete conditions are the same for both step B and C because the step following both of these is step D.

This means that if the business data events arrive in the following order:

```
Event 1: MyStatus=A, BPI_GeneratedDate=10:01
Event 2: MyStatus=D, BPI_GeneratedDate=10:03
Event 3: MyStatus=B, BPI_GeneratedDate=10:02
```

When the first event arrives the value of `MyStatus` is set to A. When the next event arrives, `MyStatus` is set to the value D - this causes both step B and step C to complete. Thus, after these three business data events have been processed, the process instance is shown to have completed steps A, B, C and D...which is not correct.

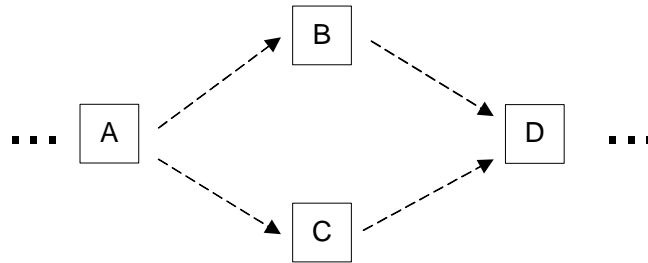
A Specific Event, or Status, for Start and Stop Conditions

Writing progression rules that can cope if business data events arrive out of sequence requires that, for each step, you have a well-defined start condition and a well-defined complete condition.

The only way to guarantee well-defined start and complete conditions is to configure a specific event, or status code, to signal the start of each step, and to configure a specific event, or status code, to signal the completion of each step.

The example process, as shown in [Figure 33](#) on page 175, uses the data attribute `MyStatus` to progress the process.

Figure 33 A Sample Process



Suppose now that `MyStatus` is populated with the values `A-Start`, `A-End`, `B-Start`, `B-End`, `C-Start`, `C-End`, `D-Start` and `D-End`.

Now you can write progression rules that have specific start and end conditions, and these progression rules will be able to cope regardless of the order in which the business events arrive.

For example, the progression rules for step `B` would be as follows:

```
Start transition:
Property: MyStatus
From:
To:      "B-Start"
```

```
Complete transition:
Property: MyStatus
From:
To:      "B-End"
```

That is, when `MyStatus` goes from any value to `B-Start`, the process has started step `B`. When `MyStatus` goes from any value to `B-End`, the process has completed step `B`.

Summary

When writing robust progression rules you need to give some thought to how the business data events are going to arrive from the underlying data systems and applications.

If you can guarantee that the business data events are going to arrive at HPBPI in correct time sequence, then your progression rules can afford to be fairly simple. For example, your start and complete progression rules can be based on specific data transitions such as “when the data attribute value transitions from `X` to `Y` you start (or complete) the step”. Another alternative

is that your start and complete progression rules could be simply based around the value of a single data attribute where you say something like “when the `MyStatus` attribute is set to `X` the process is in step `X`”.

If, however, you cannot guarantee that the business data events are going to arrive at HPBPI in correct time sequence then you need to think carefully about your progression rules. Your progression rules can not rely upon data transitions. With business data events arriving out of sequence you must receive either a specific status, or a specific event, to signal the start of a step, and a specific status, or event, to signal the end of a step.

If you are unable to receive separate events or status values within your business data events, then you could possibly consider ways to sort your business before they are sent into HPBPI, and thus guarantee the events are sent to HPBPI in the correct order.

6 Creating A Business Process

This chapter takes you through a slightly more real life and complex scenario where you define a high-level business process for an Insurance firm.

As you work through this chapter there are labs that get you defining the Insurance Claim process. In between the labs there are explanations taking your through the possible discussions/decisions that might be made when building a real-life process.

Initial Investigations

Let's consider the kind of initial investigation work that goes into defining your process.

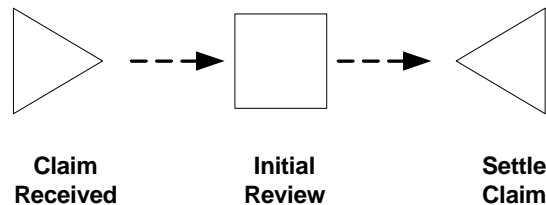
Initial Process Definition

The first task is always to gain an understanding of the underlying business process and then to sketch out the high-level HPBPI business process that you want to use for monitoring.

This initial HPBPI process should be a high-level abstraction of the underlying business process. Keeping it simple.

So the initial proposal for your high-level Insurance process is as follows:

Figure 34 Stage 1 High-Level Insurance Process



where:

- **Claim Received**

When a claim is received, the details are entered into the claims handling system. The claim is not officially recognized until the claims handling system has assigned a claims number. A claims number is not assigned until the input data has passed a number of data validation tests, for example, validating that the customer has a policy that is in date.

- **Initial Review**

When the claims system has acknowledged receipt of the claim, it is assigned to a claims handler, an individual who oversees the processing of the claim. *Initial Review* is a review by the claims handler to evaluate the nature of the claim.

- **Settle Claim**

When a claim has been accepted, a settlement letter is drafted and the payment details are transmitted to the settlement system. A record of the letter and the claim details are archived.

Initial Data Definition

You need to start defining the kind of data that your process is to use/maintain.

You should start this by identifying how it is that you know when you are in a particular step. That is, you need to understand what activities start each step and what activities signal that a step is complete.

These might include:

- When a claim number is assigned to a claim, the claim is officially recognized as started and the `Claim Received` step is started
- When a claim has been assigned to a claims handler, the `Initial Review` step is started
- When the claim has been accepted, the `Settle Claim` step is started
- When a customer letter is drafted, the `Settle Claim` step is completed

In addition to needing data to allow you to know when steps start and complete, you also need to define the data that you would like to maintain for monitoring purposes. Ask yourself the question:

“What statistics do I want to get from this process?”

You might consider the following:

- How many claims are processed
- How long it takes to accept a claim
- The value of claims accepted

More Detail Required?

At this stage you could go ahead with the current simple process definition, however, in understanding the real underlying business process you start to realize that settling a claim is not the only outcome of the Initial Review step.

You then decide that you do wish to monitor these additional outcomes and be able to answer questions such as:

- How many claims are rejected
- How long is it taking to resolve a disputed claim
- How long is it taking to obtain a medical or technical report
- When does the number of outstanding settlements exceeds a predefined threshold

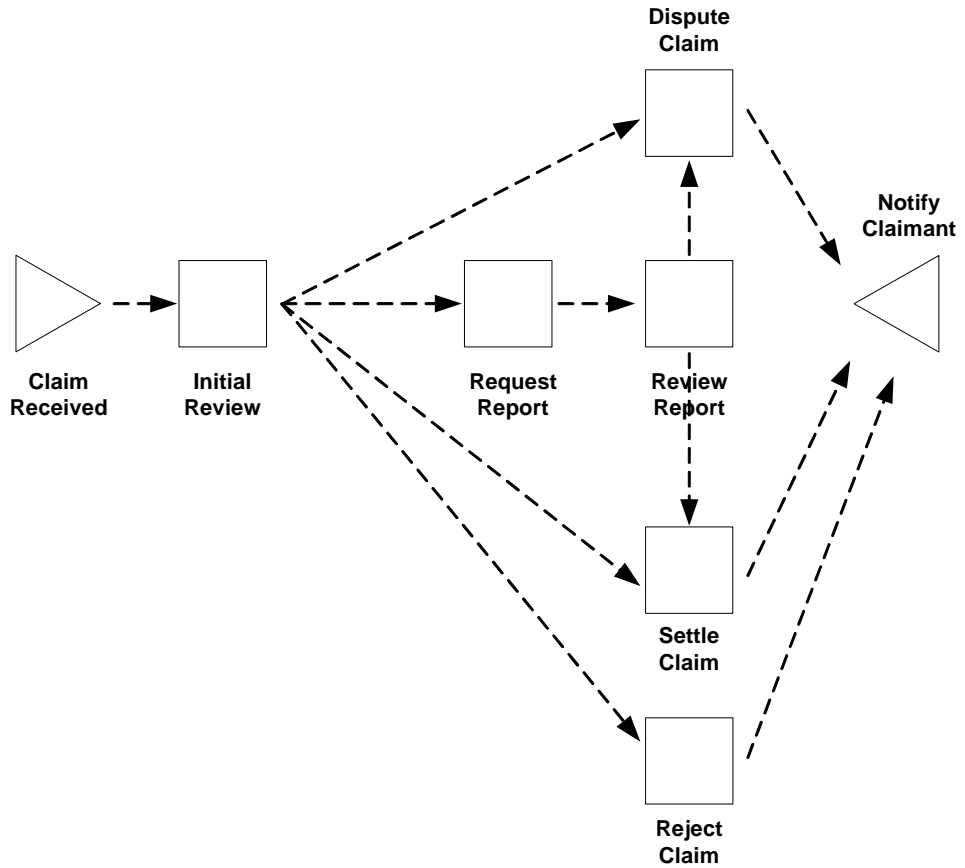
You decide that you need to expand the process definition to include more detail...

Stage Two Investigations

To monitor these additional outcomes of the Initial Review step, you need to introduce more steps in the process.

You decide that the process now look as follows.

Figure 35 Stage 2 High-Level Insurance Process



where:

- **Claim Received**

A process instance is initiated when the HPBPI Business Impact Engine receives an event from the claims handling system signaling that a claim number/identifier has been assigned.

- **Initial Review**

When the claims handler has been assigned by the claims handling system, the claim moves to the `Initial Review` step, so the start condition is the assignment of a claim handler identifier to the claim record.

The review can complete with four possible outcomes:

- the claim is rejected; this may be due to a number of technical reasons.
- the insurance company disputes the claim; and further information is required.
- the claim requires further investigation by a specialist (e.g. a doctor or mechanic).
- the claim can be settled.

Any of these four outcomes are considered a legitimate conclusion to this step and any one of them could be the completion condition for the `Initial Review` step.

- **Reject Claim**

The start condition for this step is that the insurance company has rejected the claim following an initial review. Once a claim has been rejected, a rejection letter is drafted and a record of the letter and the claim details are archived. The completion condition is when the draft letter is registered with the settlement system.

- **Dispute Claim**

The start condition for this step is that the insurance company is disputing the claim. Once a claim is in dispute, a dispute letter is drafted and a record of the letter and the claim details are archived. The completion condition is when the draft letter is registered with the settlement system.

- **Settle Claim**

The start condition for this step is that the insurance company is settling the claim. Once a claim has been accepted, a settlement letter is drafted and the payment details are transmitted to the settlement system. A record of the letter and the claim details including the settlement are archived. The completion condition is when the draft letter is registered with the settlement system.

- **Request Report**

The start condition for this step is that the insurance company is requesting a report for the claim. Once a report has been requested, a requisition letter is drafted and transmitted (faxed) to the appropriate specialist. The completion condition is when the draft letter is registered with the settlement system.

- **Review Report**

The start condition for this step is when the claims handler receives the report back from the specialist. Note that the process does not include the precise details of how the report is received or when reminders need to be sent. The acquisition of reports could be a process in its own right; here you are only interested in monitoring the high level process.

The start condition is when the claim handler receives the report. The possible outcomes of the review are that the claim is either disputed or settled. The completion condition is either when a settlement letter is drafted and sent to the settlement system, or when a dispute letter is drafted.

- **Notify Claimant**

This step enables you to measure how long it takes for customers (claimants) to be notified of the results of a claim. This step starts when the drafted claims letter is registered, and completes when the claims letter is actually sent (passed to the paper mail system).

The completion of this step results in the completion of the process.

Now you have identified the basic start/completion conditions for each step, you need to assess what other data is required in order to answer the question:

“What statistics/monitoring does the customer want to get from this process?”

Answers to questions such as:

- What are the most common paths through the process
- Where are the hold ups in the process
- What is the value of claims outstanding
- Details of the backlog of claims that are currently outstanding

The data definition side of things is considered in more detail later in this chapter, but for the moment let's go ahead and create the process definition using the HPBPI Modeler.

Lab - Defining the Basic Process Diagram

In this lab you defines the process diagram.

The Process Diagram

- Run the HPBPI Modeler
- Create a new process, called: Insurance Claim
- Draw the process so that it matches the process diagram shown in [Figure 35](#) on page 181

This is the Stage 2 Insurance process diagram - the one that includes steps for all the different outcomes such as Dispute Claim, Reject Claim, etc.

A Sequenced Arc

You want this process to send an alert if the Review Report step is started before the Request Report step (because this should never happen!), so:

- Set the arc that connects Request Report to Review Report and mark it such that its sequence is checked

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

Data Requirements

Developing the process and associated data is an iterative process and at some point you need to check that the process can be implemented, and that the data on which you plan to report can be obtained from the underlying application systems. You can check this later with the person who is integrating the business process into the operation infrastructure.

If you are unsure about what data to include in the process, think about what your measures are. Keep asking yourself the questions:

“What statistics/monitoring does the customer want to get from this process?”

“What statistics/monitoring do I want to get from this process?”

Is it the value of an order, the cumulative number of orders, or is it queue sizes to indicate orders building up? The business measurements that you want to collect dictate the data that you need in your business process.

Be aware that if you define too much data, the performance and manageability of the process are impacted, so you need to make sure that you include only the data that you need; you do not need to include all the data that is available. Too much data also creates the integrator problems when trying to obtain the data from the lower-level application systems; it also makes the process too heavyweight and affects performance. Keep in mind that you are monitoring and not trying to control the real underlying business process.

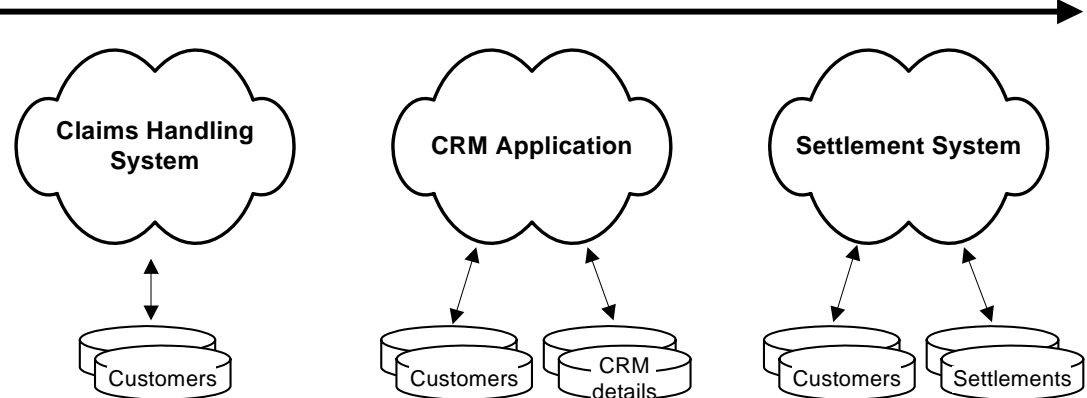
Data Normalization

The data you need to define within the HPBPI Modeler actually comes from underlying application systems through data events. That is, the data is already out there in the various application system. You do not need to redefine all of that data within the HPBPI Modeler - just the data that you would like to monitor.

You may only need to bring in key values (such as `ClaimID`, `ClaimantID`) because when you come to write statistical reports, you can use these IDs to access the underlying databases directly to retrieve the necessary additional information. That is, there is no need to hold the full personal details of each claimant within the insurance process - just the `ClaimantID` is enough.

Where this can get tricky is when your underlying business process actually moves across disparate application systems. For example:

The business operates across all of these...



All these applications, potentially, have a different view of the Customer, Claim and other data, depending on the function of the application within the business. And typically the unique ID (for example, the `ClaimID`) is not the same between each system.

To be able to model a data definition that can track a claim as it passes across these different application systems you require one or more of the following:

- You could have your applications people set up a common “overall” id for your claim that is valid across all applications. This is unlikely to happen...but could be possible
- The applications people could provide a mechanism for sending an event into the HPBPI Business Impact Engine that maps between the various key values

That is, the application systems generate events something like this:

- When a new claim is raised in the Claims Handling System, the `ClaimID` is sent in as (for example) the value `123`

- When ClaimID 123 moves across to the CRM Application, an event is generated saying that ClaimID 123 is now mapped to CRM-ClaimID ABC-45H
- When CRM-ClaimID ABC-45H moves across to the Settlement System, an event is generated saying that CRM-ClaimID ABC-45H maps to SS-ClaimID 20040319-Bent45
- The applications people could set up a system (database, application, etc.) that keeps track of all the different IDs across the three systems and allows cross reference lookups

This allows events to contain all the key values as they are available. That is, each event looks up in this cross-reference data table and adds on the current known values of all three keys.

When you define the data definition within the HPBPI Modeler you define all three IDs as being unique. That is, the claim can be uniquely identified by any one of these keys.

For situations where the data is across disparate systems, data normalization is required as you need some way to know that the same claim is known by different IDs within each of the systems. Whilst this can be time consuming to set up, at least it is only required for the actual key ID fields. There is no need to normalize all your data across all your applications - just the key ID fields such that a claim can be tracked. (This topic is also covered in [Multiple Unique IDs](#) on page 143.)

For the lab work in this chapter, you can assume that the ClaimID is unique across all underlying applications.

Let's now consider the data properties that you might maintain for your Insurance Claim process...

Data Used To Progress The Process

The first set of data that you need to define is the data used to progress the process, that is, the data that provides the start and completion conditions for each step. You did this when you were gathering information to draw out the process diagram.

Additional Data Properties

In addition to the data required to progress the process, you define the data that you need, and ensure that the required business impact and alert information is available, for example, amount(\$) of the claim, etc..

Here are the properties for the claims data definition to be used with the Insurance Claim process.

Claim ID

The claim ID is communicated in an event from the claims handling system. The event is produced when a recently entered claim has passed the data validation tests.

Claim Handler ID

When a claims handler is assigned to a claim, the claims system generates an event to give you this claims handler ID.

Claim State

The claims state attribute is required by the business process to be one of Initiated, Rejected, Disputed, Settled, or Report Requested. These states are not directly represented in the claims handling system. The claims handling system does maintain a state, but it is much more detailed in order to represent the various conditions of each state.

The state field in the underlying claims handling system can contain the following values:

- R101 - rejected for form errors
- R102 - rejected for lapsed policy
- R103 - rejected for unknown identity

- D104 - disputed, not covered by policy
- D105 - disputed, claim is covered elsewhere
- D106 - disputed, Terms and Conditions not met.
- S107 - settled, payment to follow
- S108 - settled, payment withheld
- S109 - settled, no payment due
- E110 - medical report requested
- E111 - mechanical report requested
- E112 - loss adjusters report required
- null - initial state

But the states required for the business process can be mapped from these more detailed claims handling system states.

Claim Value

Quite simply, the dollar (\$) value of the actual claim.

Claim Letter Status

The claim letter status attribute is required to show the status of the claim letter that is managed by the insurance company.

The possible value for the claim letter status attribute are as follows:

- Drafted - Letter drafted, waiting to be sent
- Sent - Letter sent to client
- Archived - Copy of letter archived after being sent to client

Claim Letter Type

The claim letter type attribute is required to show the types of letter sent by the insurance company.

The possible values for claim letter type are as follows:

- Settlement - Letter to client states that the claim is being settled
- Dispute - Letter to client states that the claim is being disputed
- Rejection - Letter to client states that the claim has been rejected

Claim Report Status

The claim report status attribute is required to show the status of the reports being requested by the insurance company as part of dealing with the claims.

The possible values for report status are as follows:

- Requested - Report has been requested
- Received - Report has been received back from medical, mechanical or loss adjustor
- Queried - Questions have been sent back to medical, mechanical or loss adjustor about the original report
- Closed - Report accepted and closed

Claim Report Type

The claim report type attribute is required to show the types of report being requested by the insurance company as part of dealing with the claims.

In the example, the report types are not used as the process does not analyze that level of detail. However, it is likely that the process may be expanded in the future to account for the report types, so they have been included for completeness.

The possible values for report type are as follows:

- Medical - Report required from medical team
- Mechanical - Report required from mechanical team
- Loss Adjustor - Report required from loss adjustor

Lab - Defining The Data/Progression Rules

In this lab you define the data for your Insurance Claim process, relate the data to the process, and then configure the progression rules for each step.

Data Definition

From within the HPBPI Modeler:

- Create a new data definition, called: Claims/Data
- Now define the data properties as follows:

Name	Type	Constraints	Unique
Claim_ID	Integer		Yes
Claim_Handler_ID	Integer		
Claim_State	String	30	
Claim_Value	Currency	USD	
Claim_Letter_Status	String	15	
Claim_Letter_Type	String	15	
Claim_Report_Status	String	15	
Claim_Report_Type	String	15	

Relate The Data To The Process

- Go ahead and relate this Claims/Data definition to the Insurance Claim process
- Select Claim_ID as the Identity
- Select Claim_Value as the Process Value

Progression Rules

Configure the following progression rules:

Claim Received

Start when:

Claim_ID is assigned a value (in other words, when Claim_ID goes from null to something).

Complete when:

Claim_Handler_ID is assigned a value.

Initial Review

Start when:

Claim_Handler_ID is assigned a value.

Complete when:

Claim_State is assigned a value.

Request Report

Start when:

Claim_Report_Status is assigned the value Requested.

Complete when:

Claim_Report_Status goes from Requested to any other value.

Dispute Claim

Start when:

Claim_State changes to a value that starts with the character D

Enter this using the advanced style of progression. The rule looks as follows:

```
this.data.Claim_State.after().starts("D")
```

Complete when:

The value of `Claim_State` currently starts with the character `D` and the `Claim_Letter_Status` becomes `Drafted` and the `Claim_Letter_Type` becomes `Dispute`.

To cope with the `Claim_Letter_Status` and `Claim_Letter_Type` properties being set within different events you would code this as follows:

```
this.data.Claim_State.starts("D")
&&
(
  ( this.data.Claim_Letter_Status.changed()
    ||
    this.data.Claim_Letter_Type.changed()
  )
  &&
  ( this.data.Claim_Letter_Status == "Drafted"
    &&
    this.data.Claim_Letter_Type == "Dispute"
  )
)
```

If you are sure that these two properties are both be updated within the one event then you can simplify this rule to become:

```
this.data.Claim_State.starts("D")
&&
( this.data.Claim_Letter_Status.after() == "Drafted"
  &&
  this.data.Claim_Letter_Type.after() == "Dispute"
)
```

For this lab assume that these properties are updated by the **one** event.

If you did code up the rule that coped with these properties possibly being set by different events, it would also work if they were updated by just a single event - it copes with either situation.

Review Report

Start when:

`Claim_Report_Status` changes to the value `Received`

Complete when:

The value of `Claim_State` changes from an `E` coding to either a `D` or an `S` coding.

Your rule is:

```
this.data.Claim_State.before().starts("E")
&&
(
  this.data.Claim_State.after().starts("D")
  ||
  this.data.Claim_State.after().starts("S")
)
```

Settle Claim

Start when:

`Claim_State` changes to a value that starts with the character `S`

Enter this using the advanced style of progression. The rule looks as follows:

```
this.data.Claim_State.after().starts("S")
```

Complete when:

The value of `Claim_State` currently starts with the character `S` and the `Claim_Letter_Status` becomes `Drafted` and the `Claim_Letter_Type` becomes `Settlement`.

Assuming that the `Claim_Letter_Status` and `Claim_Letter_Type` properties are both updated within the one event, your rule is:

```
this.data.Claim_State.starts("S")
&&
(
  this.data.Claim_Letter_Status.after() == "Drafted"
  &&
  this.data.Claim_Letter_Type.after() == "Settlement"
)
```

Reject Claim

Start when:

Claim_State changes to a value that starts with the character R

Enter this using the advanced style of progression. The rule looks as follows:

```
this.data.Claim_State.after().starts("R")
```

Complete when:

The value of Claim_State currently starts with the character R and the Claim_Letter_Status becomes Drafted and the Claim_Letter_Type becomes Rejection.

Assuming that the Claim_Letter_Status and Claim_Letter_Type properties are both updated within the one event then your rule is:

```
this.data.Claim_State.starts("R")
&&
(
  this.data.Claim_Letter_Status.after() == "Drafted"
  &&
  this.data.Claim_Letter_Type.after() == "Rejection"
)
```

Notify Claimant

Start when:

Claim_Letter_Status changes to the value Drafted

Complete when:

Claim_Letter_Status changes to the value Sent

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

Data Events

You have identified the data that your process needs in order to progress. The next step is to identify the events that carry this data.

Full details of how to build adapters and how to set up the Business Events Handler are described in the *HPBPI Integration Training Guide - Business Events*.

But for now, you need to define the events that you expect to receive from the underlying application systems.

Lab - Defining Events/Subscriptions

In this lab you define the events that will drive your Insurance Claim process and configure your data definition to subscribe to these events.

Event Definitions

From within the HPBPI Modeler, go ahead and define the following data events:

- Claims/New

Properties:

Claim_ID
Claim_Value

- Claims/HandlerAssigned

Properties:

Claim_ID
Claim_Handler_ID

- Claims/Reviewed

Properties:

Claim_ID
Claim_State
Claim_Report_Status

- Claims/ReportStatus

Properties

```
Claim_ID  
Claim_Report_Status  
Claim_Report_Type  
Claim_State
```

- Claims/LetterStatus

Properties

```
Claim_ID  
Claim_Letter_Status  
Claim_Letter_Type
```

Event Subscriptions

Go ahead and subscribe your Claims/Data definition to these events as follows:

- Claims/New subscription

- It is a specific subscription matching on Claim_ID
- The subscription assigns:

```
this.Claim_ID = event.Claim_ID;  
this.Claim_Value = event.Claim_Value;
```

- Claims/HandlerAssigned subscription

- It is a specific subscription matching on Claim_ID
- The subscription assigns:

```
this.Claim_ID = event.Claim_ID;  
this.Claim_Handler_ID = event.Claim_Handler_ID;
```

- Claims/Reviewed subscription

- It is a specific subscription matching on Claim_ID
- The subscription assigns:

```
this.Claim_ID = event.Claim_ID;  
this.Claim_State = event.Claim_State;  
this.Claim_Report_Status = event.Claim_Report_Status;
```

- Claims/ReportStatus subscription

- It is a specific subscription matching on Claim_ID

- The subscription assigns:

```
this.Claim_ID = event.Claim_ID;  
this.Claim_Report_Status = event.Claim_Report_Status;  
this.Claim_Report_Type = event.Claim_Report_Type;  
this.Claim_State = event.Claim_State;
```

- Claims/LetterStatus subscription

- It is a specific subscription matching on Claim_ID

- The subscription assigns:

```
this.Claim_ID = event.Claim_ID;  
this.Claim_Letter_Status = event.Claim_Letter_Status;  
this.Claim_Letter_Type = event.Claim_Letter_Type;
```

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

Lab - Deploy/Test the Insurance Claim Process

Now that you have defined your Insurance Claim process, let's test that it works as you expect:

- Go ahead and deploy the Insurance Claim process
- Using the HPBPI Business Process Dashboard you should now be able to see your deployed process definition
- Using the Process Simulator (contributed utility) inject the following events and watch the progression in the HPBPI Business Process Dashboard:

— **Claims/New**

```
Claim_ID:      10
Claim_Value:   22000
```

A new process instance should appear in the HPBPI Dashboard

— **Claims/HandlerAssigned**

```
Claim_ID:      10
Claim_Handler_ID: 999
```

The process instance should now move to the Initial Review step.

— **Claims/Reviewed**

```
Claim_ID:      10
Claim_State:    S107
Claim_Report_Status: %null%
```

This should progress the process to the Settle Claim step.

— **Claims/LetterStatus**

```
Claim_ID:      10
Claim_Letter_Status: Drafted
Claim_Letter_Type: Settlement
```

This should progress the process to the Notify Claimant step.

— **Claims/LetterStatus**

Claim_ID: 10
Claim_Letter_Status: Sent
Claim_Letter_Type: Settlement

This should complete the process.

- Let's try sending another claim through the system. This time going through the Request Report step...

— **Claims/New**

Claim_ID: 11
Claim_Value: 23000

— **Claims/HandlerAssigned**

Claim_ID: 11
Claim_Handler_ID: 888

— **Claims/Reviewed**

Claim_ID: 11
Claim_State: E110
Claim_Report_Status: Requested

— **Claims/ReportStatus**

Claim_ID: 11
Claim_State: E110
Claim_Report_Status: Received
Claim_Report_Type: Medical

— **Claims/ReportStatus**

Claim_ID: 11
Claim_State: D104
Claim_Report_Status: Closed
Claim_Report_Type: Medical

— **Claims/LetterStatus**

Claim_ID: 11
Claim_Letter_Status: Drafted
Claim_Letter_Type: Dispute

— **Claims/LetterStatus**

```
Claim_ID:          11
Claim_Letter_Status:  Sent
Claim_Letter_Type:  Dispute
```

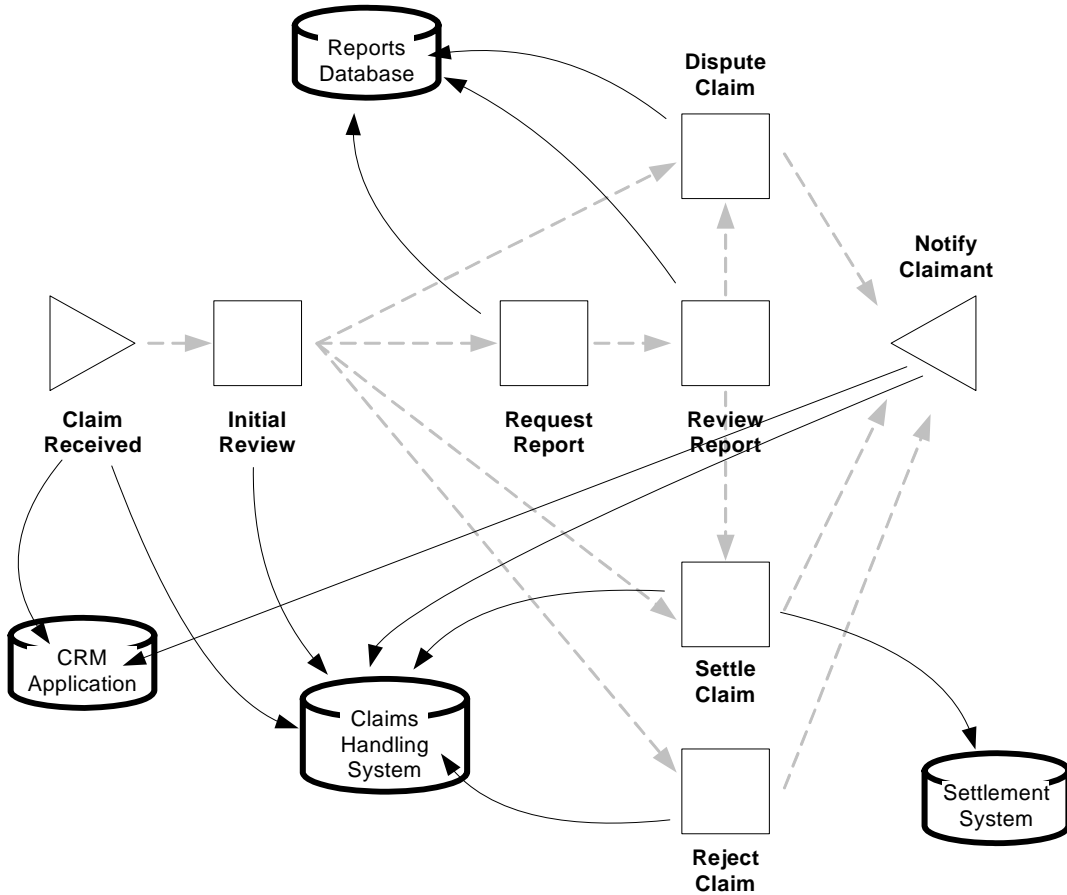
Well done! You have reached the end of the lab.

Services

Now the process is deployed you can go into Business Availability Center and identify the services that are utilized at each step of your process. A single process step could depend on one or more IT services.

For example, your insurance claim process might have the following IT dependencies:

Figure 36 Insurance Process and IT Service Dependencies



Lab - Defining IT Services Dependencies



This lab assumes that you have access to a Business Availability Center installation.

Enabling the Integration with BAC

Run the HPBPI Administration Console and enable the integration between HPBPI and your Business Availability Center installation.

Make sure you enable `Service source settings`.

Once you have applied the configuration changes, your process and its steps will exist within the uCMDB of Business Availability Center.

BAC Administration

- Logon to Business Availability Center and go to the administration tool.
- Verify that you can see the CIs for your process and its dependent steps.
- Set up either a new KPI, or an IT dependency, for at least one of your process steps.

Select a KPI or dependency where you are able to change its state.

- Change the state of your KPI, or dependency, and see the impact reflected within the HPBPI Business Process Dashboard.

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

7 Using Non-BAC Services

This chapter looks at how to link your business processes directly to your IT service hierarchy when you do not have access to HP Business Availability Center.

The preferred mechanism for HPBPI to receive IT status is from HP Business Availability Center - refer to [Chapter 3, Using BAC](#). However, if you are running HPBPI standalone from any HP Business Availability Center installation then HPBPI is able to receive IT service status events directly from the following service monitoring products:

- HP Operations Manager (HPOM)
- HP Service Oriented Architecture (SOA) Manager

This chapter looks at how to configure and use the direct integrations for HPOM and SOA Manager.

HPOM Integration

The HPBPI/HP Operations Manager (HPOM) integration basically works as follows:

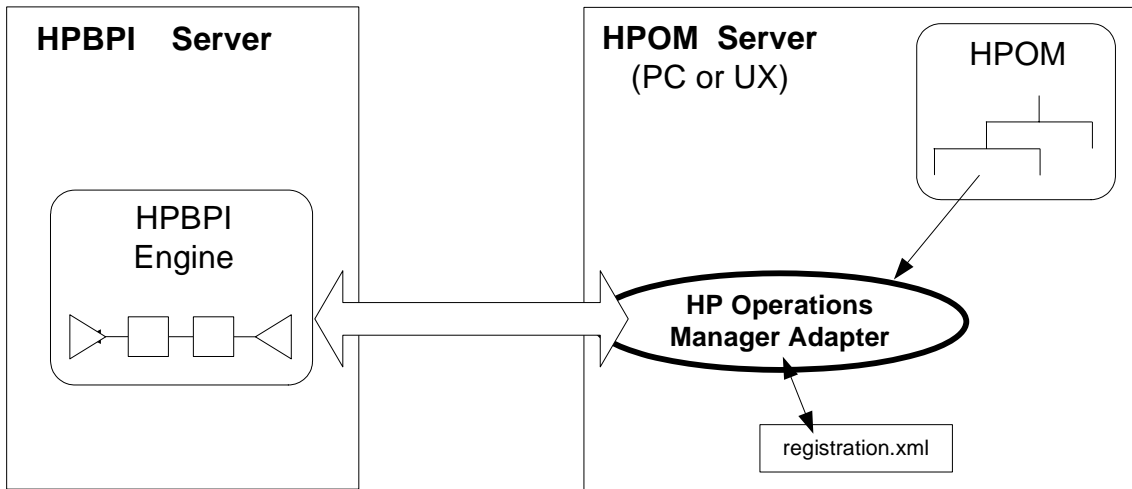
- Within the HPBPI Modeler, you are able to specify the HPOM services on which your business process depends.
- When you come to deploy this business process, HPBPI tells HPOM that it now wants to receive on-going status change information for these particular services.
- As and when these services change state, HPOM sends this status change information to HPBPI and your process reflects any resultant business impact.

Architecture

For HPOM to be able to communicate with HPBPI you need to install and run the HP Operations Manager Adapter component on the HPOM server. Refer to the *HPBPI Installation Guide* for more details.

Let's consider the architecture where you have installed HPBPI and HPOM on different servers:

Figure 37 HPOM Integration Architecture



where:

- HPOM can be installed on either UX or Windows
- All communication between HPBPI and HPOM goes through the HP Operations Manager Adapter
- Service Definition Enquiry

When defining a process within the HPBPI Modeler you specify the names of the HPOM services that you wish to use. The HPBPI Modeler uses the HP Operations Manager Adapter to determine whether these services exist within HPOM.

- Service Registration

When you deploy a process that contains references to HPOM services, the HPBPI deployer registers these service names with the HP Operations Manager Adapter. These registrations are held in the file (on the HPOM server):

```
HPBPI-install-dir/misc/bia/registration.xml
```

This enables the HP Operations Manager Adapter to know what status change events it needs to send back to HPBPI.

- Service Status Events

Once the HP Operations Manager Adapter has a list of registered HPOM services, any status changes for these services are passed through to HPBPI.

Configuring The HPBPI HPOM Integration

Configuring the HPBPI HPOM integration requires some set up on both the HPOM server and the HPBPI server. Even if you have installed HPBPI on the same server as HPOM, these steps are still required:

- On the HPBPI Server

You configure the HPOM connection details using the HPBPI Administration Console.

- In the left-hand navigation pane, select the option:

Operational Service Sources

- In the right-hand pane, double click the option:

HP Operations Manager

The main piece of information that the HPBPI server needs to know is the host name of the server running HPOM.

- Accept the other defaults and click the `OK` button.
- Ensure that your new adapter has a tick in the `Enabled?` column.
- Click `Apply` to save these configuration details.

You can refer to the *HPBPI System Administration Guide* for more details.

After configuring your HPOM connection you **need to restart all HPBPI components**.

- On the HPOM Server

You need to install the HP Operations Manager Adapter on this HPOM server. Refer to the *HPBPI Installation Guide* for more details.

Once the HP Operations Manager Adapter is installed on the HPOM server, it is ready to be used.

Use the HPBPI Administration Console (on the HPOM server) to start up the HP Operations Manager Adapter.

The HPBPI Modeler

The HPBPI Modeler lets you import service definitions from HPOM. This then allows you to define the dependencies between the steps in your processes and the underlying IT services.

The services you import into the HPBPI Modeler are effectively just pointers (or “links”) to the services defined and managed within your HPOM. That is, the services are still maintained and owned by HPOM.

Linking HPOM Services

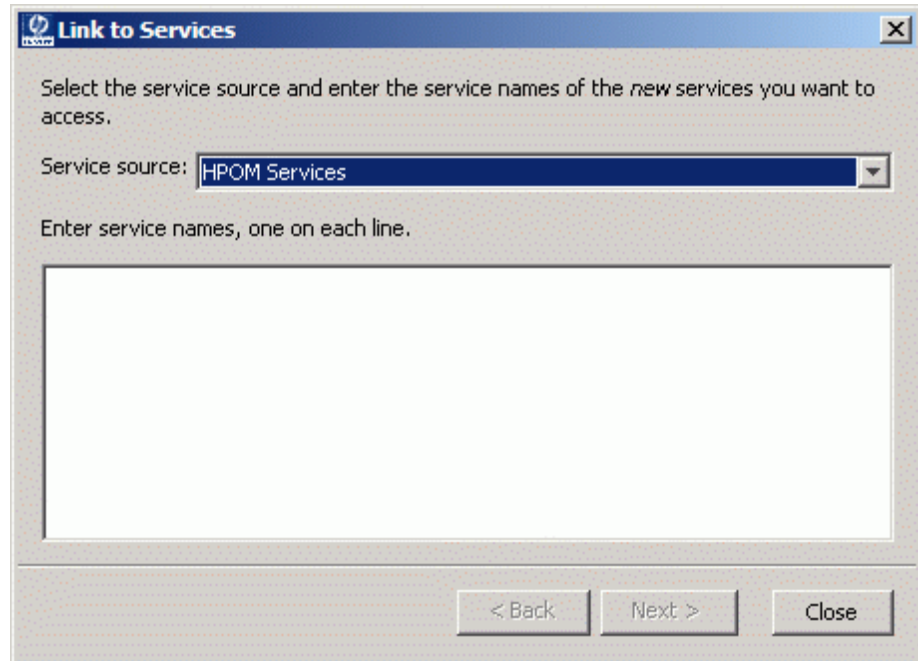
HPOM service hierarchies are typically quite large, and your business processes are typically going to depend on a small’ish subset of the full HPOM hierarchy. Hence the HPBPI Modeler does **not** import the full HPOM hierarchy. Instead, the process developer specifies the individual services by name.

Let’s walk through an example...

The steps to link HPOM service definitions are as follows:

1. Within the Modeler, select: File->Link to Services...

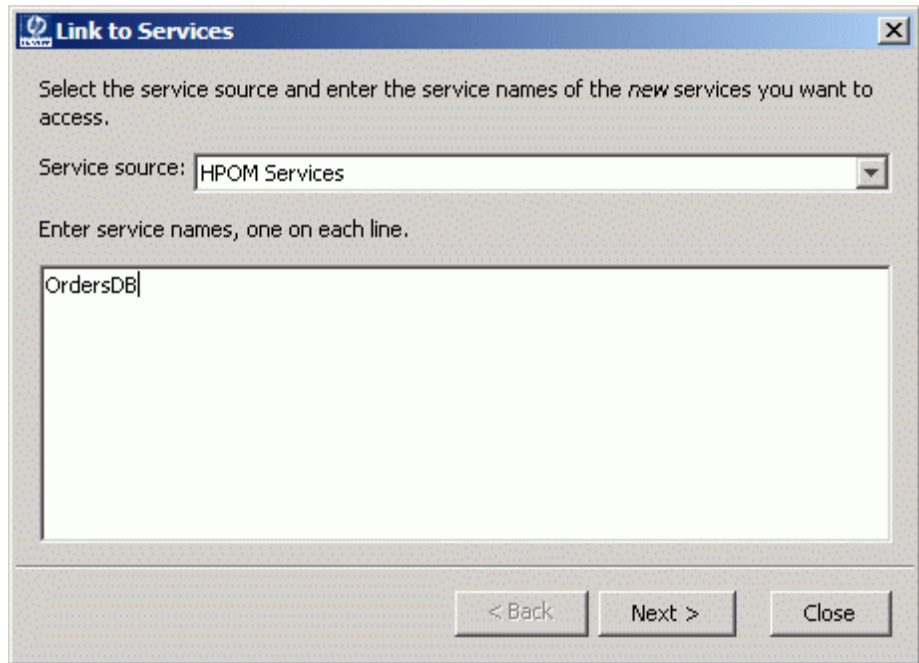
This brings up the following dialog:



2. When the dialog appears, select the option HPOM Service from the Service source drop down menu.

3. Type in the name(s) of the HPOM services you wish to link to the HPBPI Modeler

For example, to link the service definition `OrdersDB`, the dialog would look as follows:



The name(s) that you type here must match the HPOM service name(s). These are the unique names assigned when you created the service within your HPOM hierarchy. It is not the display label for the HPOM service.

If you want to link in more than one service, simply type each name on a new line.

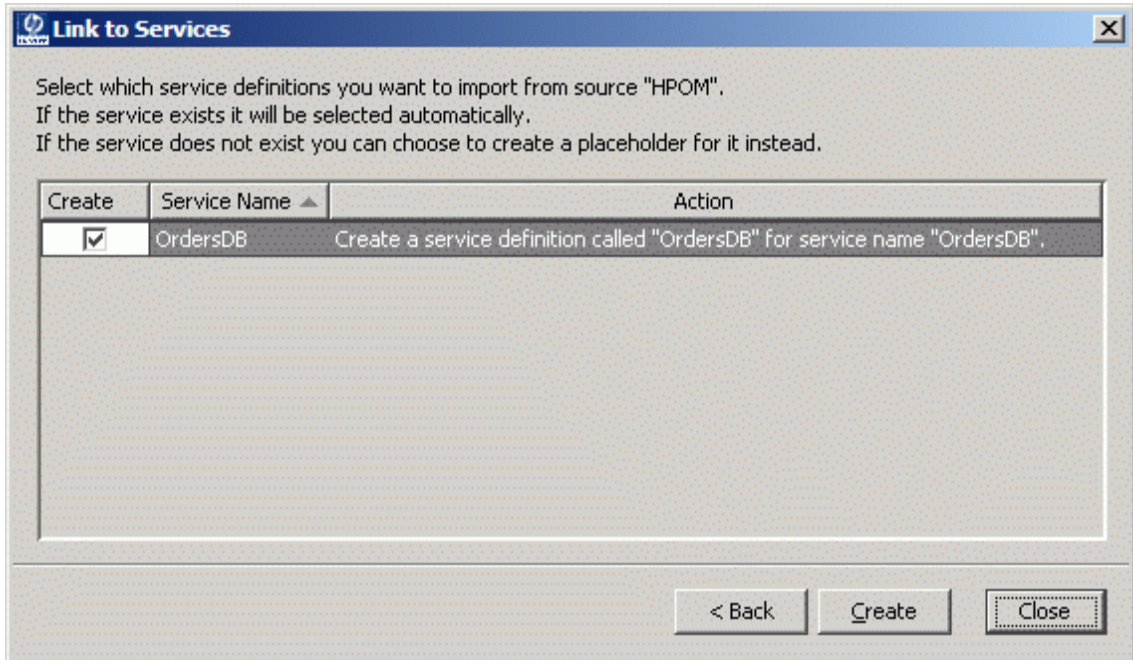
Be careful linking to HPOM services whose names differ only by case. HPBPI running on an Oracle database handles this, however, HPBPI running on a MSSQL database is not able to distinguish between them.

The HPOM service names that you are linking to must not exceed 255 characters.

4. Click on the `Next` button

It is at this point that the HPBPI Modeler goes out and talks to HPOM (using the HP Operations Manager Adapter) to locate the service definition for the name(s) specified.

If the HPBPI Modeler is able to communicate with HPOM then you see the following dialog:



If the service name exists in HPOM it is listed here with the `Create` option checked - all ready for you to accept.

If the service name does not exist in HPOM, it is listed here with the `Create` option left un-checked. This occurs for two main reasons:

- Either you mis-typed the service name

In which case you click the `Back` button and re-type the correct name.

...or...

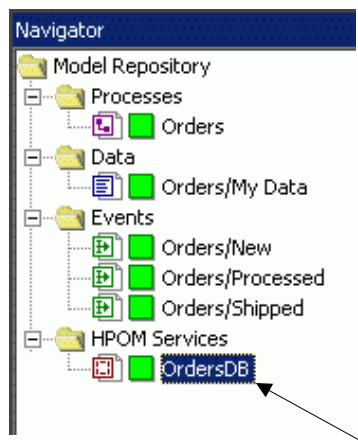
- You have specified a name that you know is going to exist within HPOM some time in the future and you want the HPBPI Modeler to create a place holder

By creating a placeholder within the HPBPI Modeler, this allows you to define business processes that use this service. When this service is eventually created within HPOM, any deployed processes start receiving status change events for this service.

5. Once the correct services are listed and you have correctly set the `Create` option for each one, simply press the `Create` button

You see a progress dialog. When this is completed, you can close this progress dialog.

You are now able to see your newly linked HPOM services. They are listed in the Navigator pane on the HPBPI Modeler, under a folder called HPOM Services.



Notice that the newly linked service `OrdersDB` is marked as deployed (with a green icon next to its name). This does not mean that it is deployed within the HPBPI Business Impact Engine, it means that this service corresponds to an active HPOM service definition.

Placeholder Services

Suppose you know that a service is going to be defined in HPOM sometime next week, but you want to start defining the business process right now. You want to define a placeholder HPOM service.

Consider the example where you want to define a placeholder service called: ABC Service:

- As described in the previous section ([Linking HPOM Services](#) on page 209) you select:

File->Link to Services...

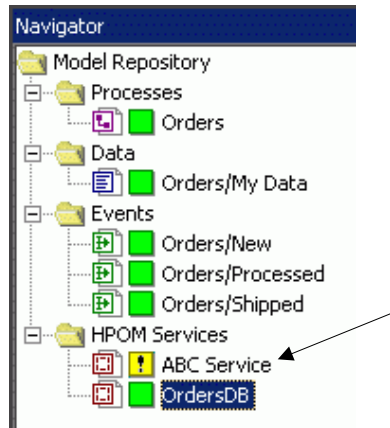
- Select the Service source called: HPOM Services
- Enter the name: ABC Service, and click Next

Notice that the Create option is not checked, and the Action column explains that if you do check the create option then this creates a placeholder for this service definition.

- Check the Create option for this service
- Now click the Create button

This creates a placeholder in the HPBPI Modeler for this service.

When you look in the Navigator pane of your Modeler, you see your service as follows:



Notice it is listed with an icon that indicates that this service definition does not currently exist in HPOM.

You can quite happily refer to a placeholder service definition within your process and then deploy this process to the HPBPI Business Impact Engine. It just means that your process does not receive any service status change events as the service does not yet exist within HPOM. Once the service has been created within HPOM your deployed process automatically starts receiving service status change events and your process reflects the business impact.

Synchronizing Services

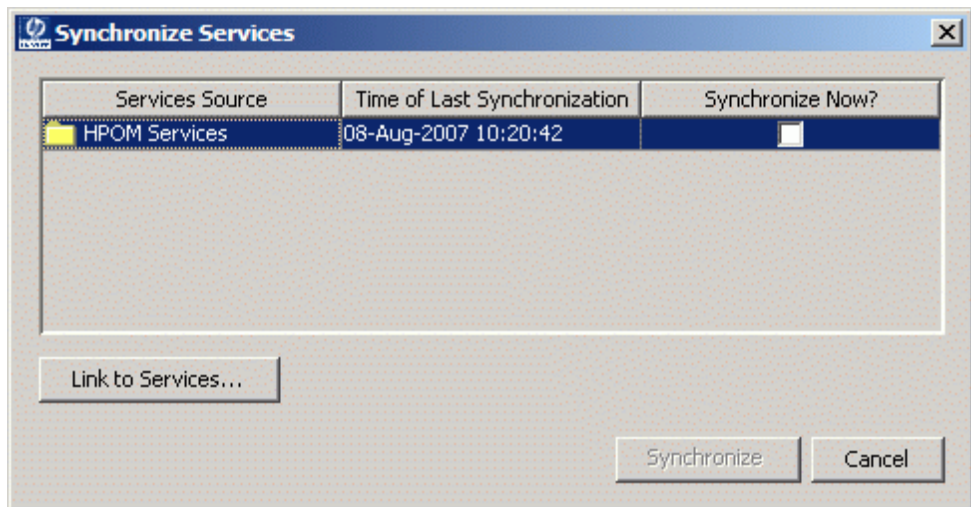
Within the HPBPI Modeler, there is the option to synchronize your service definitions. This tells the HPBPI Modeler to synchronize its view of the HPOM services.

To synchronize service definitions, you:

1. Select the menu option:

File->Synchronize Services...

You see a dialog similar to the following:



If your HPBPI installation has other service integrations configured then you also see those listed in the Services Source column.

2. You simply check the `Synchronize Now?` checkbox and click on the `Synchronize` button.

This tells the HPBPI Modeler to check the existence of all the references it has to HPOM services and see if they exist in the HPOM hierarchy:

- Any placeholder services become marked as deployed (green icon) if the service definition is found to exist in HPOM
- Any currently deployed services become placeholders if (for some reason) they have been removed from HPOM

In the example above, with the placeholder called `ABC Service`, the icon changes to green once a service of this name is created within HPOM and you have run a `synchronize`.

Notice that the `Synchronize Services` screen also has a button for linking to the services. This button takes you to the same place as the `File->Link to Services...` menu option.

Using Services Within A Process

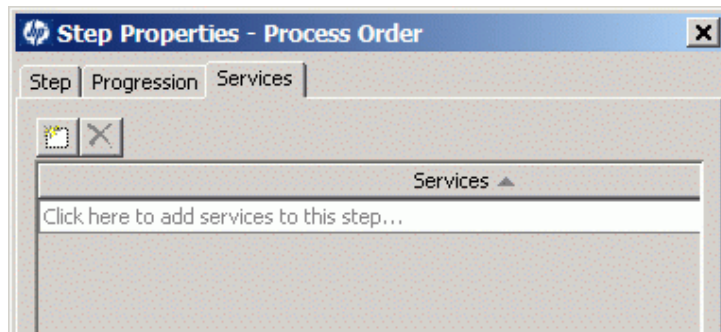
Once the HPBPI Modeler knows about some HPOM services, you can use these within your business processes.

Let's walk through an example:

Within the HPBPI Modeler you have linked in some HPOM services.

In the `Orders` process, you want the step called `Process Order` to be dependent on the HPOM service called `OrdersDB`.

- In the process editor, double-click on the `Process Order` step to bring up the `Step Properties` dialog
- Click on the `Services` tab and you are presented with a new dialog as follows:



- Click the icon

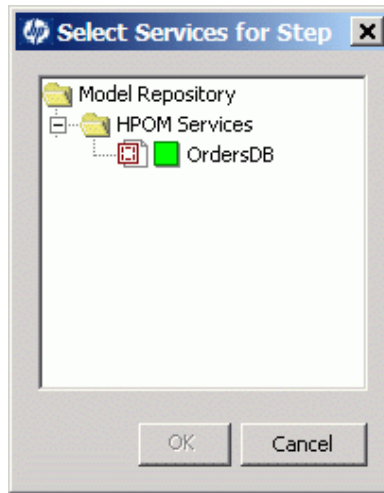


...or... click where it says to add new services.

- This displays the list of available services

This includes all services known to the HPBPI Modeler, not just the HPOM services.

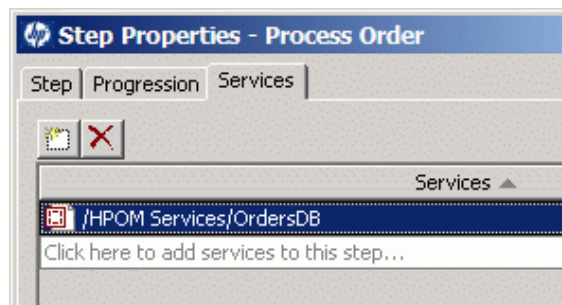
For example, the service list might look as follows:



- You then select the `OrdersDB` service and press OK

You can select multiple services. A step can be dependent on more than one service and they can be a mixture of HPOM and other services.

- The Step Properties dialog now displays the chosen service, as follows:



- You then click Close and the Process Order step is now marked as being dependent on the `OrdersDB` HPOM service

The step's tooltip within the HPBPI Modeler Process Editor pane shows this dependency. Just hold the cursor over the step image and wait for the tooltip to appear.

SOA Manager Integration

The HPBPI/SOA Manager integration basically works as follows:

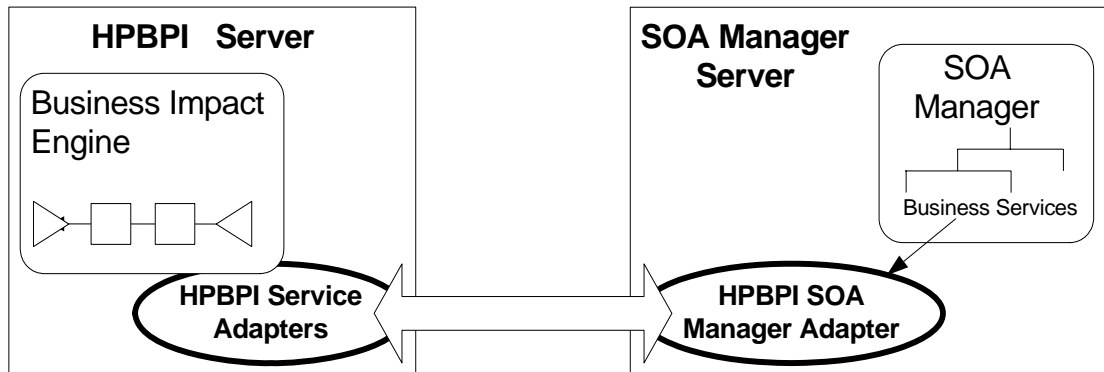
- Within the HPBPI Modeler, you are able to specify the SOA Manager Business Services on which your business process depends.
- You then deploy your process to the HPBPI Business Impact Engine.
- As these Business Services change state, this status information is collected from SOA Manager and your HPBPI process reflects any resultant business impact.

Architecture

For SOA Manager to be able to communicate with HPBPI you need to install and run the HPBPI SOA Manager Adapter component. This SOA Manager Adapter component can be installed on any machine that has access to your network, but typically you would install the SOA Manager Adapter component on the SOA Manager server. Refer to the *HPBPI Installation Guide* for more details.

Let's consider the architecture where you have installed HPBPI and the SOA Manager adapter on different servers:

Figure 38 SOA Manager Integration Architecture



where:

- You install and configure the HPBPI SOA Manager Adapter to talk to your SOA Manager installation.

- Within the HPBPI Administration Console, you configure a connection to your HPBPI SOA Manager Adapter.
- You make sure the HPBPI Service Adapters component is running, as this is the component that handles the HPBPI side of the SOA Manager connection.
- When defining a process within the HPBPI Modeler you specify the names of the SOA Manager Business Services that you wish to use. The HPBPI Modeler calls the HPBPI Service Adapters component (which, in turn, calls the HPBPI SOA Manager Adapter) to determine whether these services exist as Business Services within SOA Manager.
- You deploy your process to the HPBPI Business Impact Engine.
- The HPBPI Service Adapters component polls the HPBPI SOA Manager Adapter to get status details about the SOA Manager Business Services. Any status changes for these Business Services are passed through to the HPBPI Business Impact Engine.

Configuring the HPBPI SOA Manager Integration

Configuring the HPBPI SOA Manager integration requires some set up on both the SOA Manager server and the HPBPI server. Even if you have installed HPBPI on the same server as SOA Manager, these steps are still required:

- On the HPBPI Server

You configure the SOA Manager connection details using the HPBPI Administration Console:

- In the left-hand navigation pane, select the option:

Operational Service Sources

- Click the `Add...` then configure a connection to a SOA Manager adapter.

The main pieces of information that HPBPI needs to know are the name of the server running the HPBPI SOA Manager adapter, and the port number for the Web services communication.

You give your SOA Manager connection a name. For example: `My SOAM Adapter`. This name is then used to create the service folder within the HPBPI Modeler. Once set, you cannot modify this folder name.

- Ensure that your new adapter has a tick in the `Enabled?` column.
- Click `Apply` to save these configuration details.

You can configure connections to more than one SOA Manager adapter, you just need to give each connection a unique name.

You can refer to the *HPBPI System Administration Guide* for more details.

After configuring your new SOA Manager adapter(s) you **need to restart all HPBPI components**.

- On the SOA Manager Server

You need to install the HPBPI SOA Manager Adapter and configure it to talk to your SOA Manager installation.

Once the HPBPI SOA Manager Adapter is installed and running, it is available to HPBPI.

Refer to the *HPBPI Installation Guide* for more details.

The HPBPI Modeler

The HPBPI Modeler lets you import Business Service definitions from SOA Manager. This then allows you to define the dependencies between the steps in your processes and the underlying SOA Manager Business Services.

The services you import into the HPBPI Modeler are effectively just pointers (or “links”) to the Business Services defined and managed within your SOA Manager. That is, the services are still maintained and owned by SOA Manager.

Linking SOA Manager Services

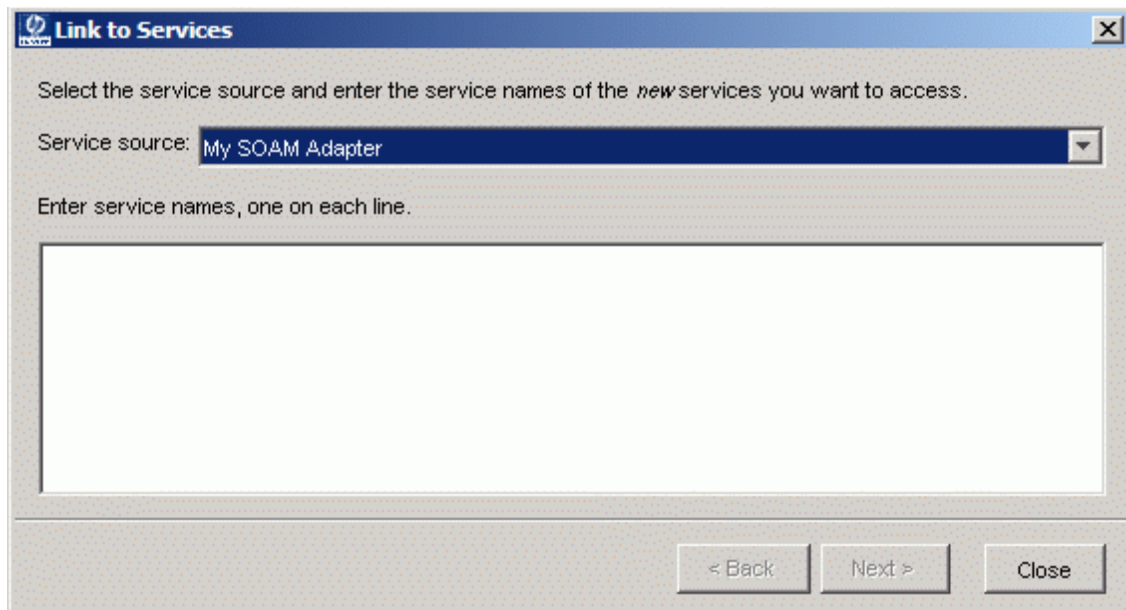
The HPBPI Modeler does **not** import the full SOA Manager Business Service hierarchy. Instead, the process developer specifies the individual services by name.

Let’s walk through an example...

The steps to link SOA Manager Business Service definitions are as follows:

1. Within the Modeler, select: File->Link to Services...

This brings up the dialog:



2. When the Link to Services dialog appears, you select the appropriate Service source. In this example, you would select the SOA Manager service source called: My SOAM Adapter. (You had previously configured this service source using the HPBPI Administration Console. See [Configuring the HPBPI SOA Manager Integration](#) on page 221)
3. Type in the name(s) of the SOA Manager services you wish to link to the HPBPI Modeler. The service names must be Business Services as defined within SOA Manager. (Within SOA Manager, you can use the Business Service Explorer (BSE) to list the available Business Services.)

For example, to link the SOA Manager Business Service definition Finance, the dialog would look as follows:

The screenshot shows a dialog box titled "Link to Services". The instructions inside the dialog are: "Select the service source and enter the service names of the new services you want to access." There is a "Service source:" label followed by a dropdown menu that currently shows "My SOAM Adapter". Below that, it says "Enter service names, one on each line." and there is a large text input area where the word "Finance" has been typed. At the bottom of the dialog, there are three buttons: "< Back", "Next >", and "Close".

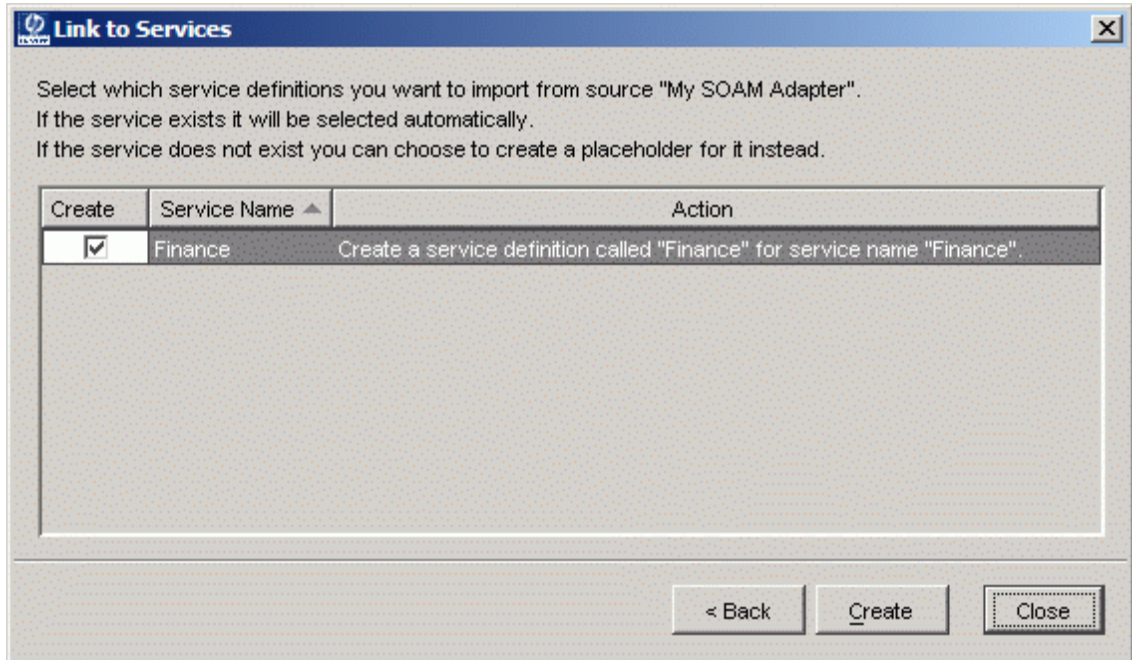
The name(s) that you type here must match the name of the Business Service(s) within SOA Manager.

If you want to link in more than one Business Service, simply type each name on a new line.

4. Click on the `Next` button

It is at this point that the HPBPI Modeler goes out and talks to the SOA Manager adapter to locate the Business Service definition for the name(s) specified.

If the HPBPI Modeler is able to communicate with the SOA Manager Adapter then you see the following dialog:



If the service name exists in SOA Manager as a Business Service, it is listed here with the `Create` option checked - all ready for you to accept.

If the service name does not exist in SOA Manager, it is listed with the `Create` option left un-checked. This occurs for two main reasons:

- Either you mis-typed the service name

In which case you click the `Back` button and re-type the correct name.

...or...

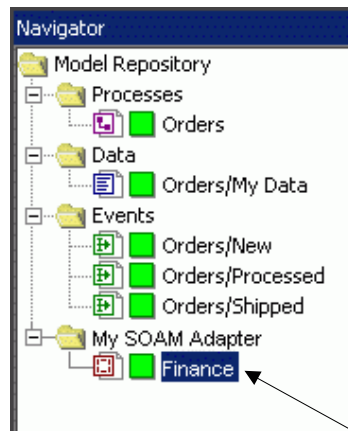
- You have specified a name that you know is going to exist within SOA Manager some time in the future and you want the HPBPI Modeler to create a place holder.

By creating a placeholder within the HPBPI Modeler, this allows you to define business processes that use this service. When this service is eventually created within SOA Manager, any deployed processes start receiving status change events for this service.

5. Once the correct services are listed and you have correctly set the `Create` option for each one, simply press the `Create` button.

You see a progress dialog. When this is completed, you can close this progress dialog.

You are now able to see your newly linked SOA Manager Business Services. They are listed in the Navigator pane of the HPBPI Modeler, under a folder called: `My SOAM Adapter` - which is the name of the service source you used to import this service.



Notice that the newly linked service `Finance` is marked as deployed (with a green icon next to its name). This does not mean that it is deployed within the HPBPI Business Impact Engine, it means that this service corresponds to an active SOA Manager Business Service definition.

Placeholder Services

Suppose you know that a service is going to be defined in SOA Manager sometime next week, but you want to start defining the business process right now. You want to define a placeholder SOA Manager Business Service.

Consider the example where you want to define a placeholder service called: ABC Service:

- As described in the previous section ([Linking SOA Manager Services](#) on page 222) you select:

File->Link to Services...

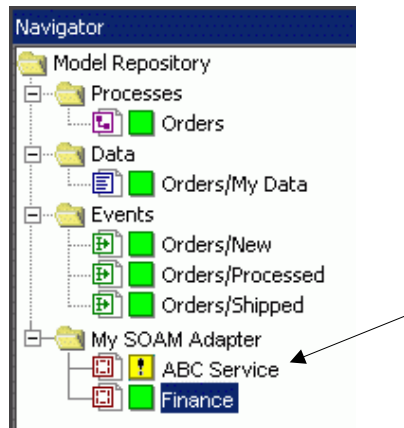
- Select the appropriate service source. For this example, you select the SOA Manager adapter configured as: My SOAM Adapter.
- Enter the name: ABC Service, and click Next

Notice that the `Create` option is not checked, and the `Action` column explains that if you do check the create option then this creates a placeholder for this Business Service definition.

- Check the `Create` option for this service
- Now click the `Create` button

This creates a placeholder in the HPBPI Modeler for this Business Service.

When you look in the Navigator pane of your Modeler, you see your service as follows:



Notice it is listed with an icon that indicates that this service definition does not currently exist in SOA Manager.

You can quite happily refer to a placeholder service definition within your process and then deploy this process to the HPBPI Business Impact Engine. It just means that your process does not receive any service status change events as the Business Service does not yet exist within SOA Manager. Once the Business Service has been created within SOA Manager your deployed process automatically starts receiving service status change events and your process reflects the business impact.

Synchronizing Services

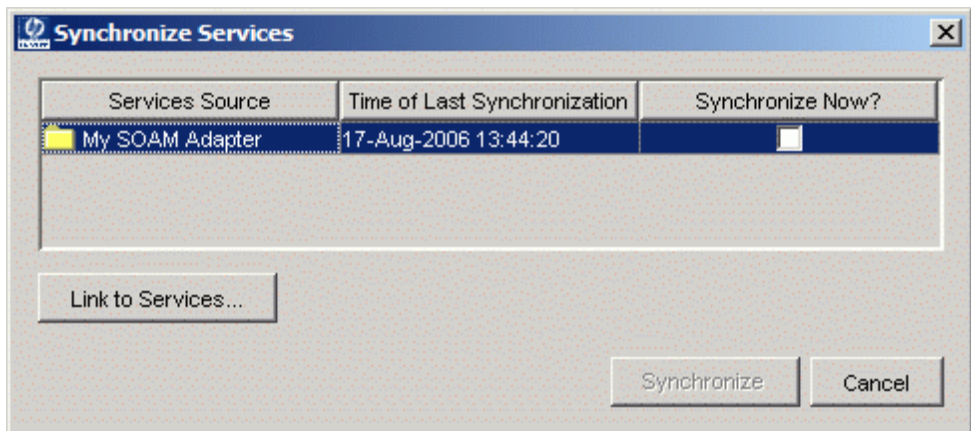
Within the HPBPI Modeler, there is the option to synchronize your service definitions. This tells the HPBPI Modeler to synchronize its view of the SOA Manager Business Services.

To synchronize service definitions, you:

1. Select the menu option:

File->Synchronize Services...

You see a dialog similar to the following:



If your HPBPI installation has other service integrations configured then you also see those listed in the Services Source column.

2. You simply check the `Synchronize Now?` checkbox and click on the `Synchronize` button.

This tells the HPBPI Modeler to check the existence of all the references it has to SOA Manager Business Services and see if they exist in the SOA Manager hierarchy:

- Any placeholder services become marked as deployed (green icon) if the Business Service definition is found to exist in SOA Manager
- Any currently deployed services become placeholders if (for some reason) they have been removed from SOA Manager

In the example above, with the placeholder called `ABC Service`, the icon changes to green once a Business Service of this name is created within SOA Manager and you have run a synchronize.

Notice that the `Synchronize Services` screen also has a button for linking to the services. This button takes you to the same place as the `File->Link to Services...` menu option.

Using Services Within A Process

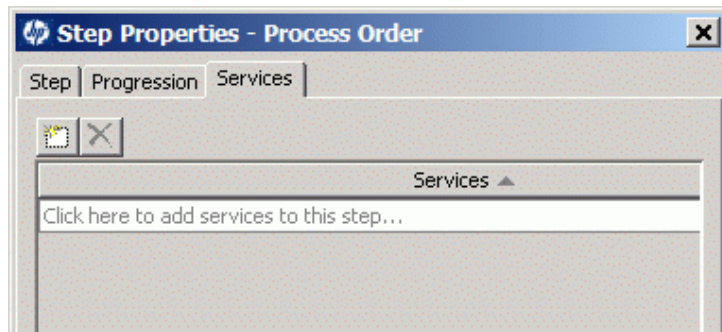
Once the HPBPI Modeler knows about some SOA Manager Business Services, you can use these within your business processes.

Let's walk through an example:

Within the HPBPI Modeler you have linked in some SOA Manager Business Services.

In the `Orders` process, you want the step called `Process Order` to be dependent on the SOA Manager Business Service called `Finance`.

- In the process editor, double-click on the `Process Order` step to bring up the `Step Properties` dialog
- Click on the `Services` tab and you are presented with a new dialog as follows:



- Click the icon

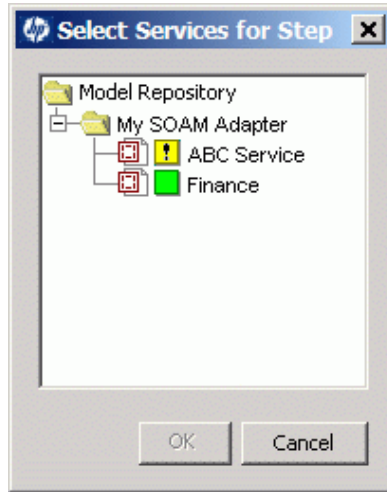


...or... click where it says to add new services.

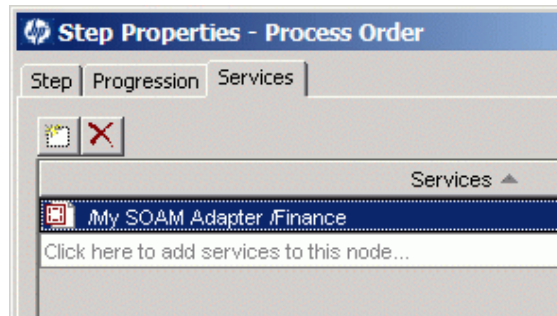
- This displays the list of available services

This includes all services known to the HPBPI Modeler, not just the SOA Manager Business Services.

For example, the service list might look as follows:



- You then select the `Finance` service and press `OK`
You can select multiple services. A step can be dependent on more than one service and they can be a mixture of SOA Manager and other services.
- The Step Properties dialog now displays the chosen service, as follows:



- You then click `OK` and the `Process Order` step is now marked as being dependent on the `Finance` SOA Manager Business Service

The step's tooltip within the HPBPI Modeler Process Editor pane shows this dependency. Just hold the cursor over the step image and wait for the tooltip to appear.

Standalone Services

What if you have services that are not monitored by any of the standard integrations offered by HPBPI? That is, you have services that are monitored by some 3rd party system or a new monitoring system that you are yet to write(!). How can you set these up and define processes that depend on their status?

The HPBPI Modeler lets you define Standalone Services. That is, services that are not monitored by any of the standard integrations offered by HPBPI.

Defining a New Service

Within the HPBPI Modeler, you can define a standalone service by either selecting:

File->New->Standalone service

or simply click on the new standalone service icon:

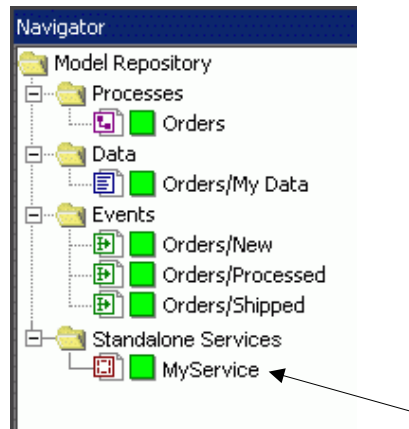


Create a new
standalone service...

This brings up the service definition page - in the right hand pane of the Modeler.

You can then assign the standalone service a name and (optionally) a description.

Your new service is then listed in the Navigator pane, under the Standalone Services folder. For example, if you create a standalone service called MyService, it might appear as follows:



Using a Standalone Service

Once your standalone service is defined within the HPBPI Modeler, you are able to specify your steps as being dependent upon this service in exactly the same way as you would for an HPOM or SOA Manager service.

You can then deploy the process.

For the HPBPI Business Impact Engine to receive service status events for this standalone service you must set up some form of monitoring system (external to HPBPI) that monitors the actual service and then sends in service status change events through the HPBPI Event Handler. Refer to the *HPBPI Integration Training Guide - Business Events* for details on how to set this up.

