HP Business Availability Center

for the Windows and Solaris operating systems

Software Version: 8.00

Business Process Insight Integration Training Guide
Modeling Processes

This guide is not being updated for Business Process Insight, from versions 8.0x and later.

Document Release Date: January 2009
Software Release Date: January 2009

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.

Third-Party Web Sites

HP provides links to external third-party Web sites to help you find supplemental information. Site content and availability may change without notice. HP makes no representations or warranties whatsoever as to site content or availability.

Copyright Notices

Trademark Notices

Adobe® and Acrobat® are trademarks of Adobe Systems Incorporated.

Intel®, Pentium®, and Intel® Xeon™ are trademarks of Intel Corporation in the U.S. and other countries.

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

Microsoft®, Windows®, Windows NT®, and Windows® XP are U.S registered trademarks 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 guide'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

This site requires that you register for an HP Passport and sign-in. To register for an HP Passport ID, go to:

http://h20229.www2.hp.com/passport-registration.html

Or click the **New users - please register** link on the HP Passport login page.

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:

http://www.hp.com/go/hpsoftwaresupport

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

HP Software Support Online provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the HP Software Support web 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:

http://h20230.www2.hp.com/new_access_levels.jsp

To register for an HP Passport ID, go to:

http://h20229.www2.hp.com/passport-registration.html

Contents

1 Introduction

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

Overall Architecture

You can think of a running BPI Server 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 BPI Modeler and is then deployed to the Process Repository.

- The 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 Process Repository, 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 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 and so on. They need to be configured by someone who understands the underlying data sources and the BPI Event Handler. Configuration these event feeds is covered in detail in the *BPI Integration Training Guide - Business Events*.

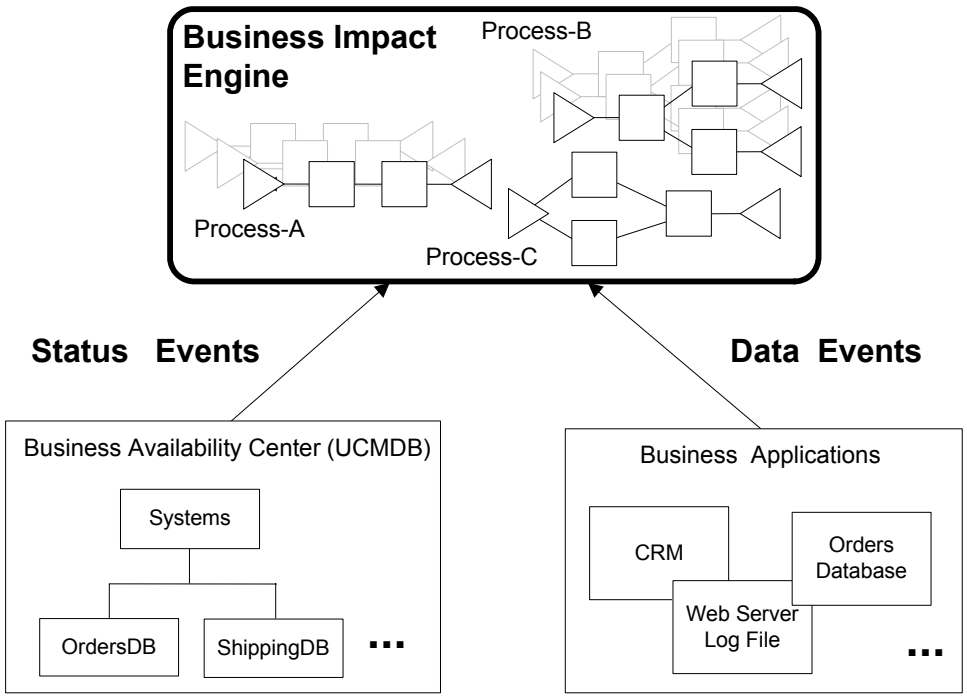
- The IT Infrastructure Status Feeds

  Once you have configured the 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 BPI.

  You can configure the Business Impact Engine to take direct “status feeds” from HP Business Availability Center. Then, if there is an IT operational resource failure, the 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.

A running BPI system might look something like Figure 1.

Figure 1 High-Level View of Components



In Figure 1:

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

  The status feeds are coming from the Universal CMDB (Configuration Management Database) within 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.

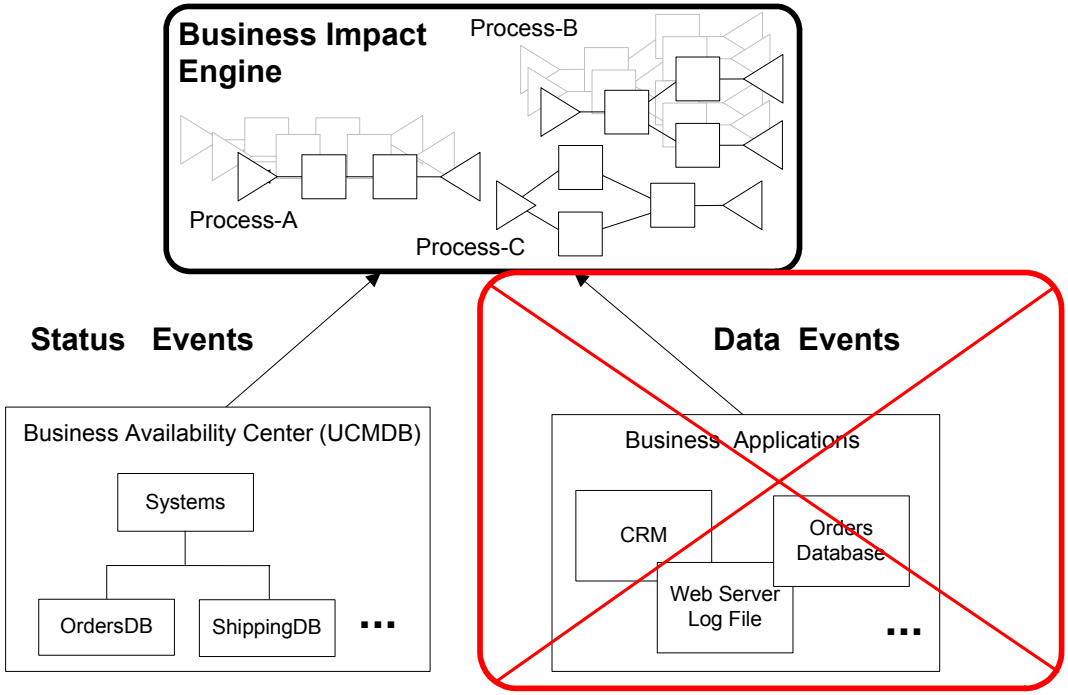
  At any one time the Business Impact Engine may be maintaining many parallel instances of any deployed process model.

BPI Server Installation

When you first install Business Availability Center (BAC), a subset of the BPI components are also installed with BAC. In this configuration, you can model a Business Process and map individual process Steps to the CIs representing the underlying IT infrastructure (IT operational resources). When you deploy this type of Business Process, you can then use the Dashboard Application to view the status of the health of the IT operational resources from the Flow Map Tab.

You cannot model Data and Event definitions unless you install the BPI Server and these features are disabled within the BPI Modeler. You can only map Steps to CIs; see Figure 2.

Figure 2 Modeling with No BPI Server Installed



More information about using BAC when there is no BPI Server installed can be found in *Reference Information for Business Process Insight*. The remainder of this document assumes that you have a fully installed and functioning BPI Server installation and that you have all of the features of the BPI Modeler available to you.

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`

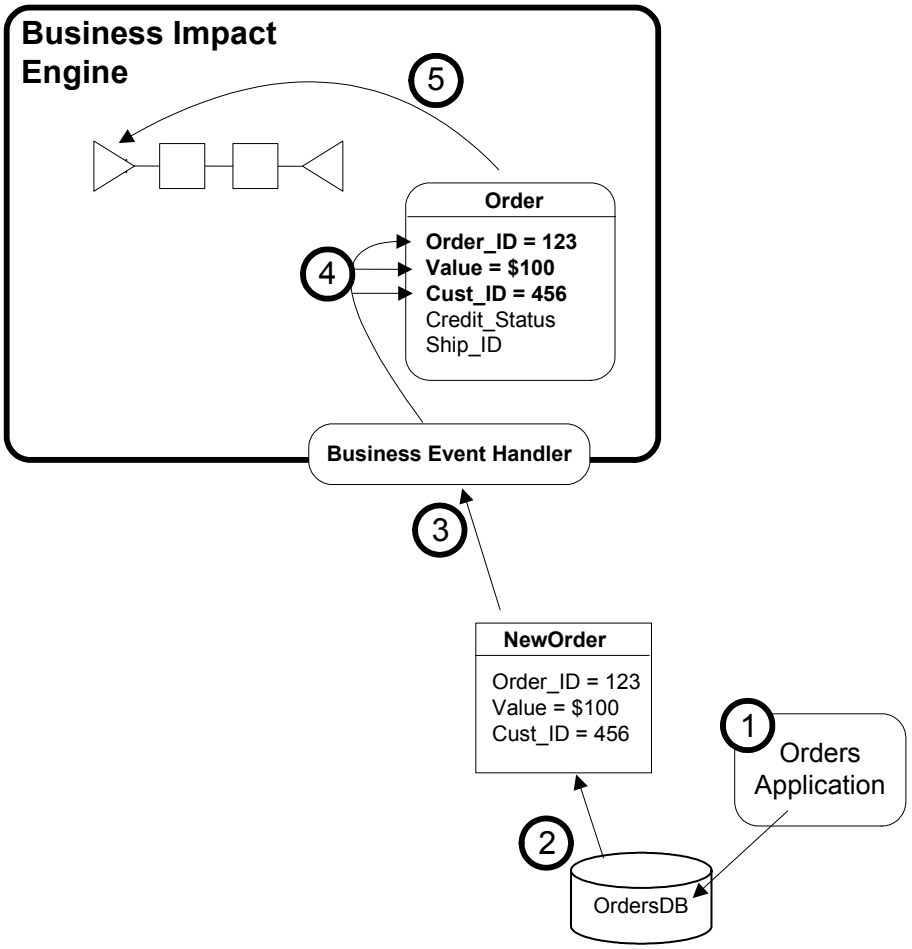
  The ID assigned by the shipping system when the order is shipped.
- You have then defined how the process knows where it is, based on values within this set of data attributes. (These are called **progression rules**.)

  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 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 3 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 Business Impact Engine to say that a new order has been placed.

   You would like to send an event into the Business Impact Engine specifying the order number (in this case `123`), the value of this order (for example: `$100`), and the customer ID for the customer who 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 Business Impact Engine as a properly formed “event”. Full details on how to configure such adapters are found in the *BPI 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 Business Impact Engine (passing through the BPI 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 Business Impact Engine to monitor each order individually. Thus the 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 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.

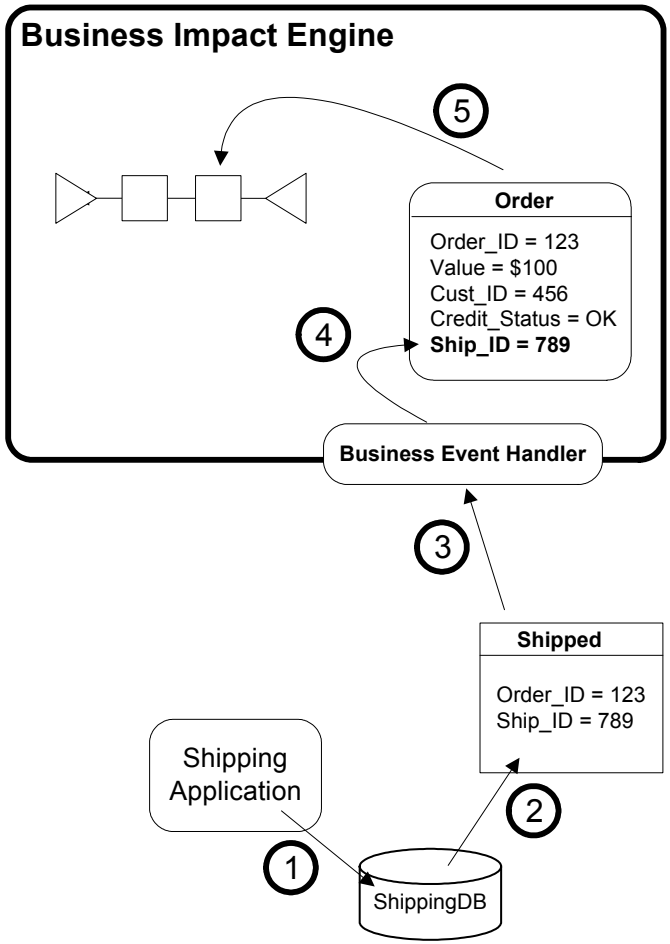
   For example, in Figure 3 the progression rules would simply say that when a new order comes in, the process instance is positioned at the first Step. In other words, when a new order comes in, the first Step within the process instance is considered “active”.

Hopefully you can see that using the BPI Server you can design a process model and the 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 4 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 adapter 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 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 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 Business Impact Engine monitoring the orders as they move through your business. This is excellent! You can now use the 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.

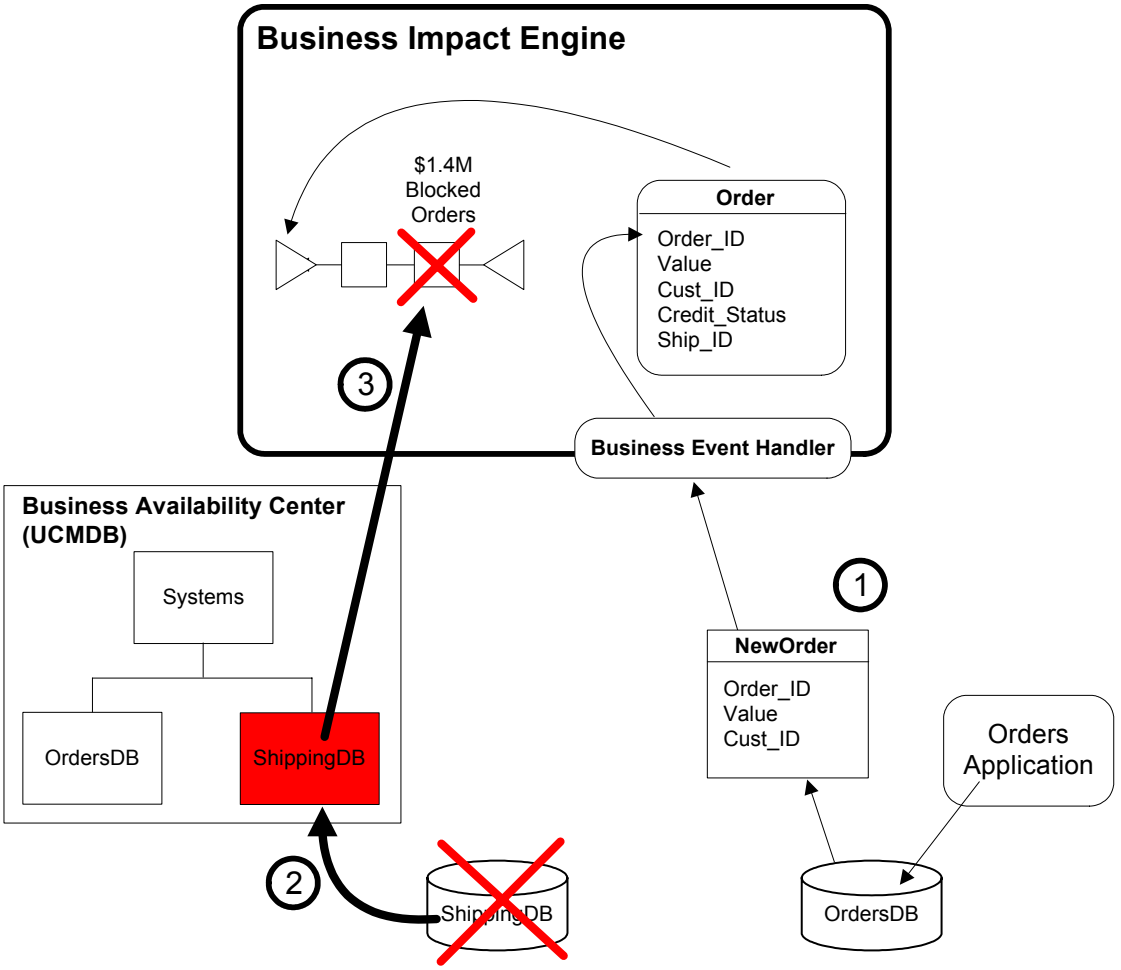
But you can also take this one step further...

Adding Service Status Feeds

The 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 5 Receiving Service Status Feeds



where:

1. New orders are being entered into the system and monitored by the 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 IT operational resource status feeds you are not only able to have BPI monitor your business volumes and throughput, but you are able to report actual business impact when any underlying IT operational resources 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 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 Business Impact Engine
- Monitor the Business Impact Engine statistics

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

Monitoring Your Process Definitions

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 operational resource becoming unavailable. The question is: “How do you see all this?”

BPI Application

The BPI Application is a component of BAC and is where the Business Process Health data is presented. The information presented on the BPI Application includes a `Scorecard`, `Health` and `Reports` page. These enable you to view each process that is deployed to the Business Impact Engine, drill down to see each instance of a process and see reports based on historical data that has been collected for the Business Process. It also shows you the associated process instance data and impact information.

You access the BPI Application as follows:

```
Application > Business Process Insight
```

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 BPI 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, you need to consider the following:

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 a BPI data event.

3. Whether or not the incoming events will be arriving in sequence?

   If you know that business data events will arrive out of sequence, you need to set up your progression rules in a specific way to allow for this. Section Out of Sequence Business Data Events on page 147 provides details of how to do this and you are strongly advised to read this section when you come to design your progression rules.

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.

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 BPI 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 BPI *Integration Training Guide - Business Events*.

5. **Deploy the process and test it**

   Use the BPI Modeler to deploy the process definition into the Business Impact Engine. You can use tools such as the Process Simulator (contributed utility) and the BPI Application 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 Operational Resource Dependencies

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

All your IT infrastructure is modeled within the UCMDB of Business Availability Center. You need to make sure that the IT dependencies for your Business Process are configured, as CIs, within the UCMDB.

As part of modeling your Business Process, you can then link these CIs to Steps in the Business Process.

When you deploy your process, the Business Process and Business Process Steps are also stored as Configuration Items in the UCMDB; there is a CI for your process and this CI contains dependent CIs for each of the Steps of the process. As a result, the dependencies between the CIs can be monitored and BAC notifies the BPI Server of the status of the IT operational resource CIs at regular intervals.

So all your IT dependencies are modeled within the UCMDB.

2 Process Definition

This chapter looks at the BPI 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 want 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 steps and departments involved in handling an order. But for monitoring purposes you might 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 BPI is that it is not actually running your business, it is monitoring it. So if you create a process within BPI 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, 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. On the other hand, you might find that the process does indicate some issues and you decide that it is worth investing time to extend the Business Process definition and gather more information about your business.

The key here is that your Process Definition should start out simple - with only a few steps - and if it turns out that you need to gather more detailed information...you can! The Business Process can always be extended; 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 BPI business event adapters, which are to generate the data events to drive how this process is monitored, need to find out all the possible places where these business events might come from. It is a good idea to make notes about the possible data sources/ systems for each Step within your Process Definition.

Another important step is to keep asking yourself "What data do you want to get from BPI when this high-level Business Process is deployed and running?" This helps you to decide what data you want to configure the Business Process to maintain for each instance, and also helps to define the 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 data/information 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 might be able to provide the Process Definition as a BPEL export file. The BPI Modeler is able to import BPEL processes and create a corresponding BPI Process Definition from the imported BPEL.

Importing the BPEL process can provide you with a good starting point when trying to create the high-level Business Process within BPI. 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 BPI 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 30.

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

Starting the BPI Modeler

You start the BPI Modeler as follows:

1. Select `Admin > Business Process Insight`

   You are presented with a tabbed page that includes a tab for `Modeling`.

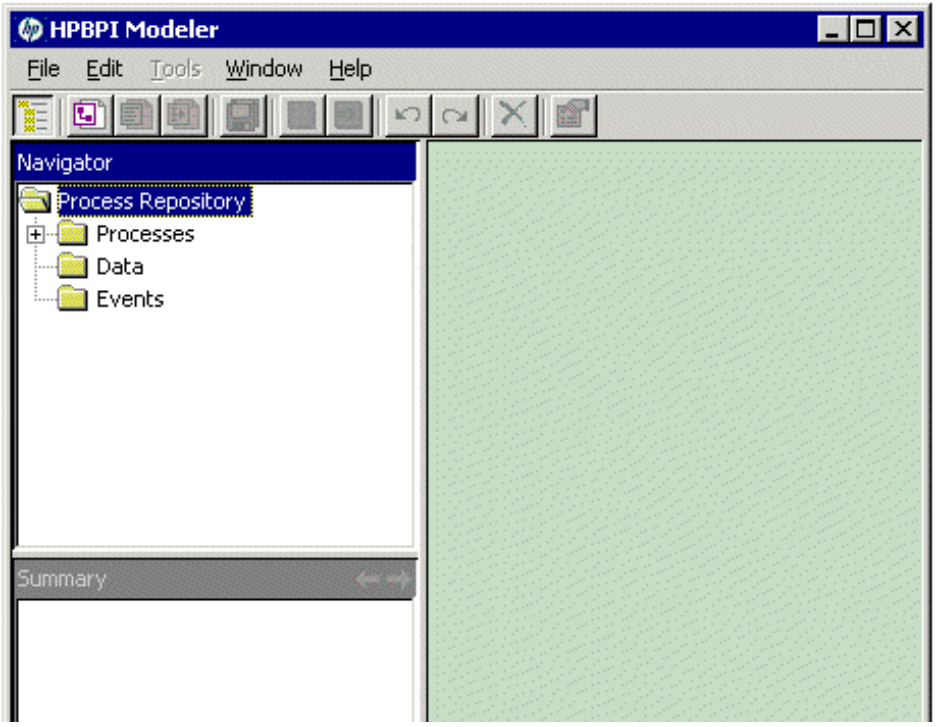
2. Select `Modeling` to open the Modeler page.

3. Click the icon for the Modeler on the page and wait for the Modeler to launch as a Java Web Start application.

   You might be asked to confirm that you want to download the files to open the Modeler within your web page. If required, confirm that you want the download to proceed.

   The BPI Modeler opens and presents a screen something like this:

Figure 6 The BPI Modeler



If you have previously defined some processes then these are listed in the Navigator pane (on the left hand side). In addition, if you have installed the BPI Server component, you also need to make sure that the `Process Repository` component is started on the BPI Server system and the BAC Infrastructure settings are completed for BPI.

The Process Repository

When you create new processes, these are saved in the Process Repository.

The Process Repository is where all of the definitions associated with your Business Processes are maintained; this includes the definitions for your processes, plus the associated data definitions and events definition, in fact, anything that you create in the BPI Modeler.

You specify the details of the Process Repository database when you install and configure BAC. The installation creates a set of data tables whose names all start with `Repos`.

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

Single User Repository

Although multiple developers can view the Process Repository, BPI currently only supports only one developer editing a Business Process using the Modeler at any one time. If you have multiple users writing to the Process 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 BPI Server, you can start to design (model) your Business Process.

Creating A New Process

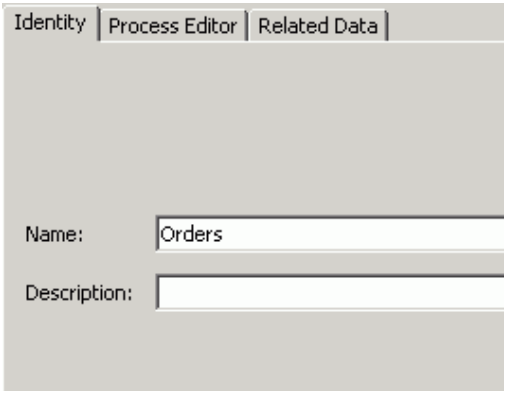
To create a process, you can either select

`File->New->Process`

or simply click on the new process icon:

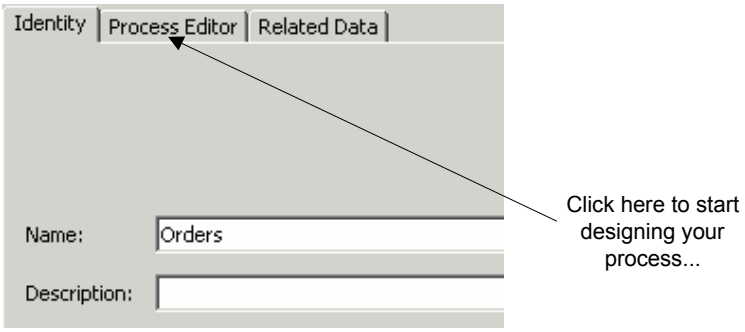
This brings up the process properties page - in the right hand pane of the BPI Modeler.

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

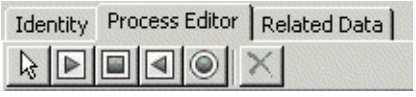


Drawing the Steps Of The Process

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:

	**Draw a Start Step** 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. 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.
	**Draw an Activity Step** 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.
	**Draw a Junction Step** 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.
	**Draw an End Step** 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.
	**The Selection Tool Icon** Clicking this icon switches you out of “Step drawing” mode and allows you to draw arcs between Steps and define Step properties.
	**Delete Steps or Arcs** Clicking this icon deletes the currently highlighted object(s).

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 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 BPI Modeler. In this instance, Junction Steps are used to help show the structure of the underlying Business Process being represented by the created BPI 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

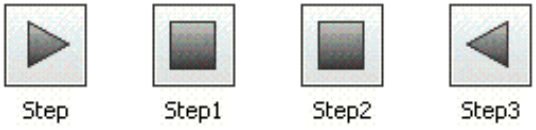
To create an End Step:

- 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

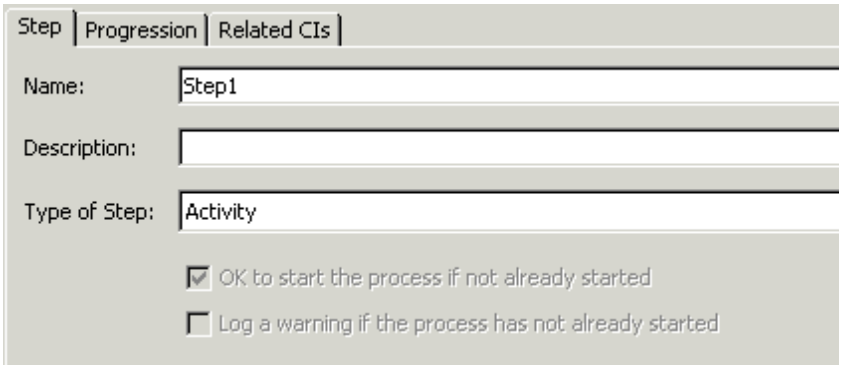
Naming Each Step

The BPI Modeler assigns default names for any newly added Steps.

To change the name of a Step you can do one of the following:

- Right-click on the Step and select `Properties`
- Double-click on the Step

The following dialog is presented:



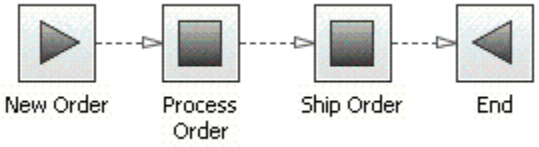
You can then enter the name of the Step, and (optionally) a description.

The other tabs on this dialog allow you to set some important things such as the progression rules and add IT operational resource dependencies (CIs) 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 BPI 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

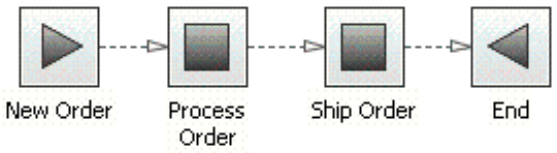
In this lab you create a process, draw some Steps and then link them with arcs.

Start up BPI

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

Start the BPI Modeler

- Start up the BPI 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 BPI Modeler's main toolbar.

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

Configuring Related CIs

The next step is to link individual Steps in the Business Process to IT operational resource CIs that have been defined within the UCMDB.

If the process has not yet been deployed, CIs can be selected for Business Process Steps, but the CI links are not synchronized to the UCMDB until the process is deployed.

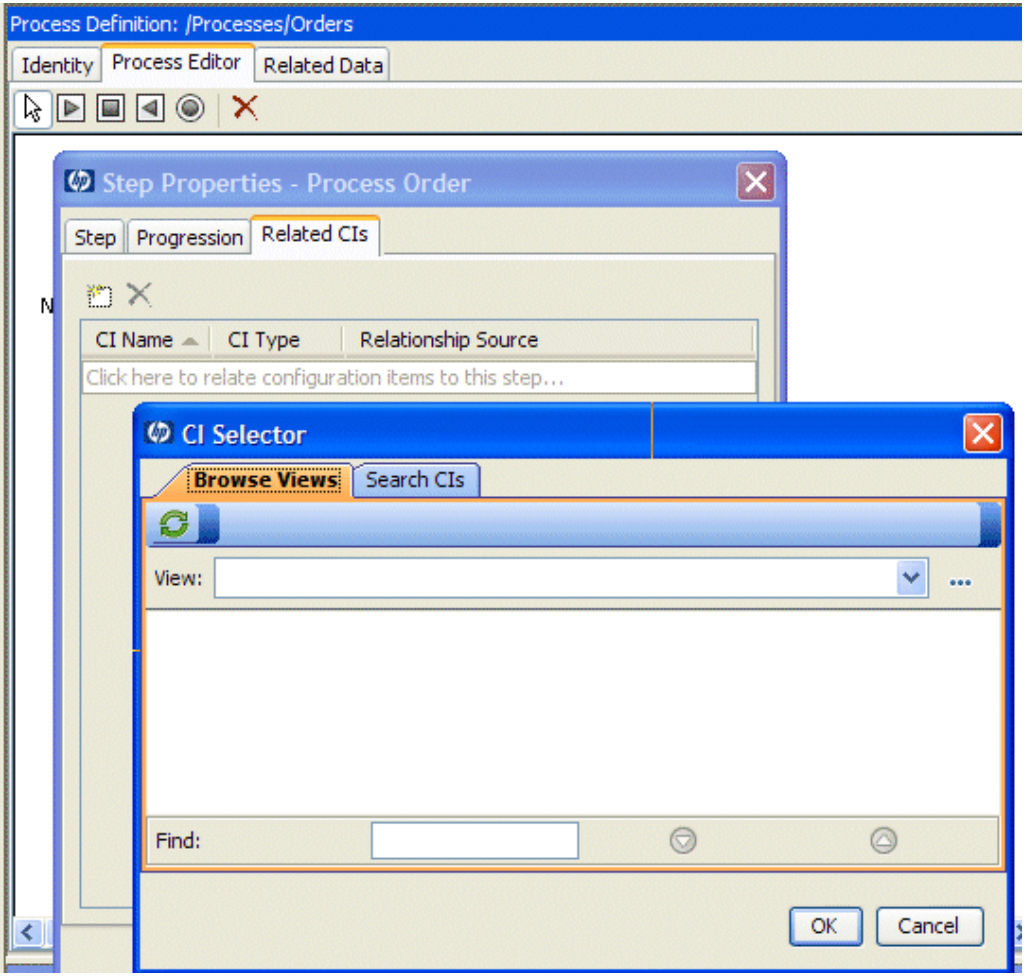
If the process has been deployed, the CI links are synchronized with the UCMDB as soon as the Business Process changes are saved.

To create a Related CI:

1. Open the Step Properties dialog.
2. Select the Related CIs tab.
3. Click under the CI Name to create a new Related CI for the Step.

   You are presented with the `CI` Selector to select views and to locate CIs as shown in Figure 7.

Figure 7 CI Selector for Related CIs



Using the CI Selector, you can select a view and browse through the list of CIs for the operational resource that you are linking to the Step.

Alternatively, you can use the `Find` option to search for a CI if you do not know in which view the CI is included.

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 BPI 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 select:

`File > New > Data`

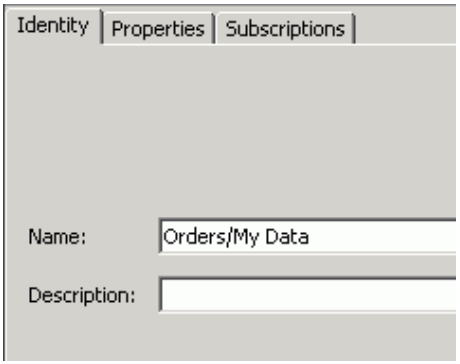
or simply click on the new data icon:

This opens up the data definition dialog in the right-hand pane of the BPI 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".

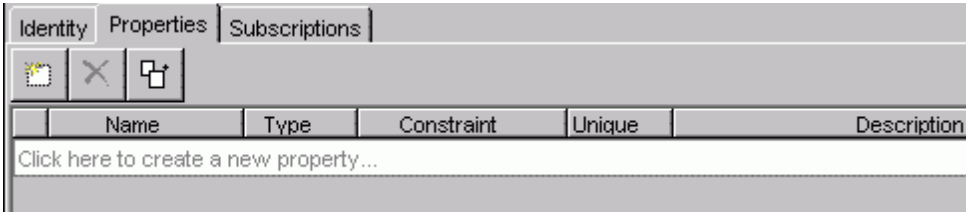
For example, you might name the data definition `Orders/My Data`, as follows:



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:

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

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 122 for further discussions.

- **Description**

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

Selecting 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 information 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!

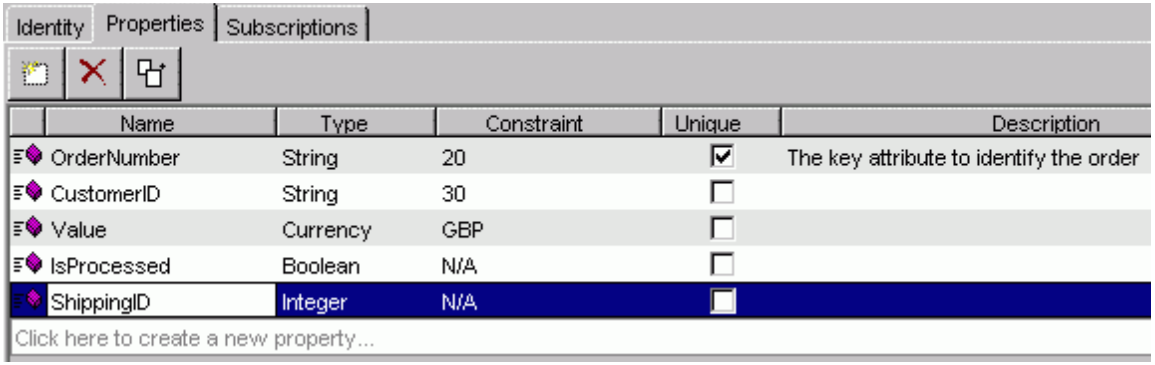
In other words, the data properties that you define provide information to enable the process to know where it is at any one time, and to collect information for later reporting/statistics.

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:



- 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:

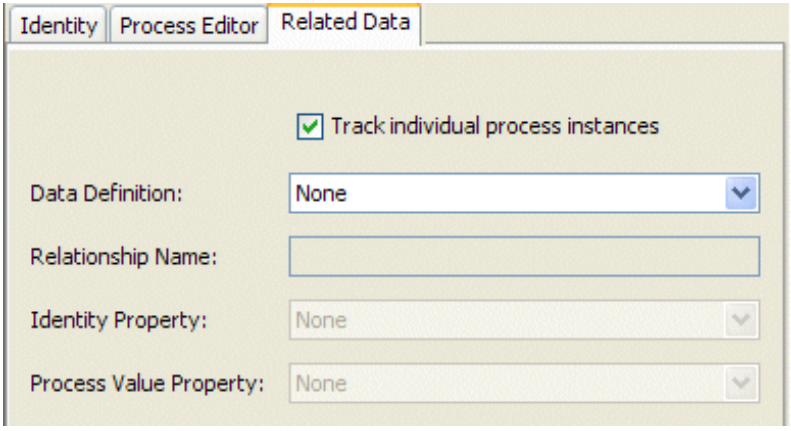
  — Select `Window` and then select the process that you wish to work on

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

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

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

  The following dialog is presented:



The following elements are included:

- **Track individual process instances**

  You cannot select this option unless you have installed a BPI Server.

  If have installed the BPI Server, you can choose to select or clear the `Track individual process instances` option as required.

- **Data Definition**

  Select the data definition for this Business Process from the drop-down list.

- **Relationship Name**

  An optional name for this relationship. This name is for your use only, so enter a name of your choice, or select the value offered.

- **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 view the Health of your Business Processes using the BPI Application Health pages. It is the property that is shown with each process instance in order that you can identify it.

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

- **Process Value Property** (optional)

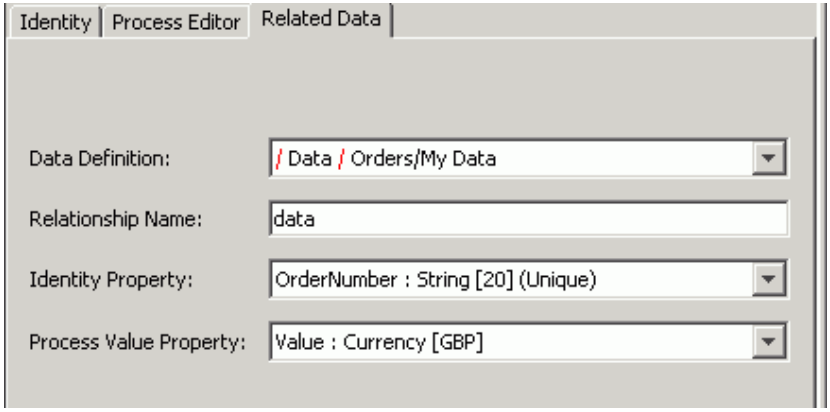
  The process value property is like the Identity property in that it is only used when you report on the process. The BPI Application displays the process value property with each process instance.

  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.

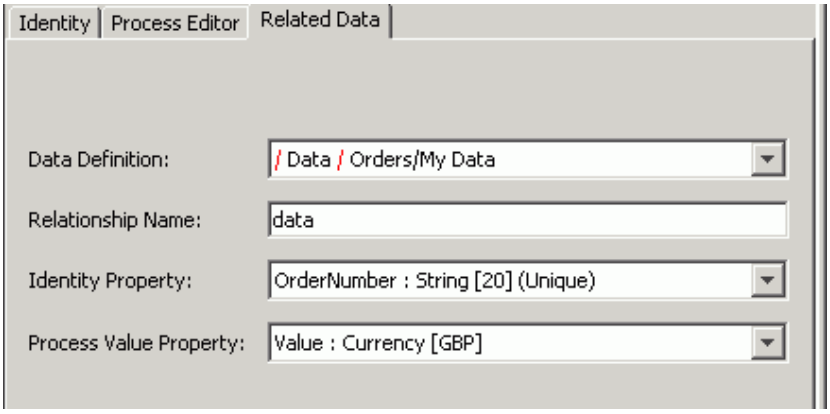
As an example, you might set up the related data as follows:



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:



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 BPI 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 *Business Process Insight Reference Information*.

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 BPI 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:

- `Start and complete on TransactionVision events`

  If the arrival of a particular event is all that is needed to progress a Business Process, then this information can be obtained directly from the business event. Using this option, you can select from a list of previously defined business events, and progress your Business Process Steps based on the arrival of a particular Business Event.

  This style of progression rule can be used in any scenario and not only with TransactionVision.

  You can also add progression rules that progress based on data within the Business Event; this is done using the Advanced Conditions option as described in Chapter 5, Advanced Progression Rules.

- `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 of the `Complete on transition` style:

  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 3:

  ```
  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 if the `Start and complete on transition` style:

  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 BPI 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.

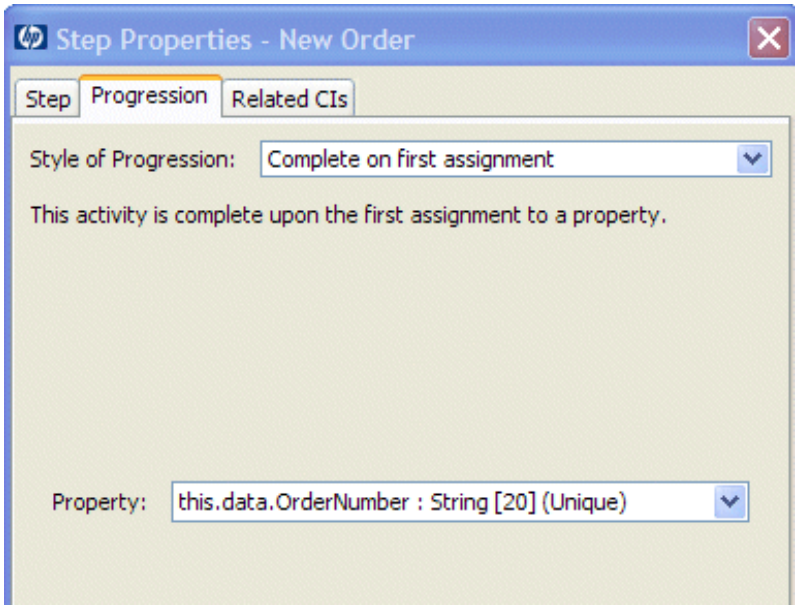
This is something that you need to think carefully about when defining your process.

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

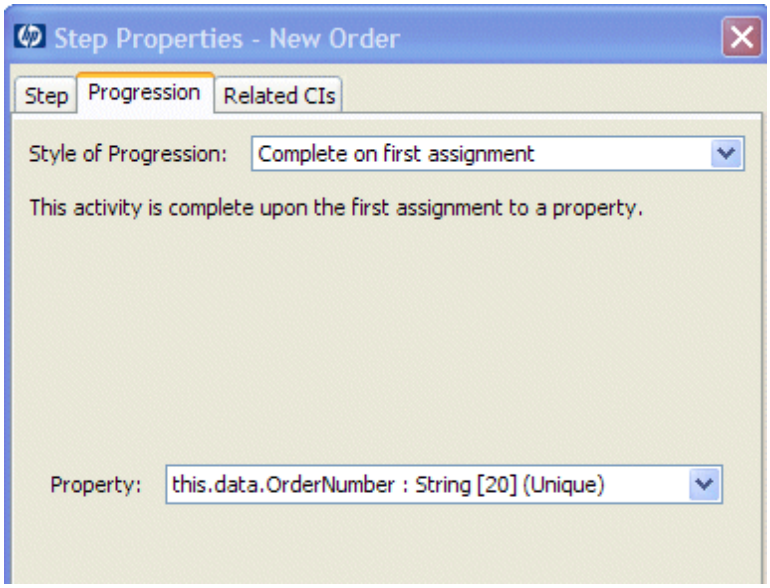
Let's configure the progression rules for your `Orders` process:

- 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 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.

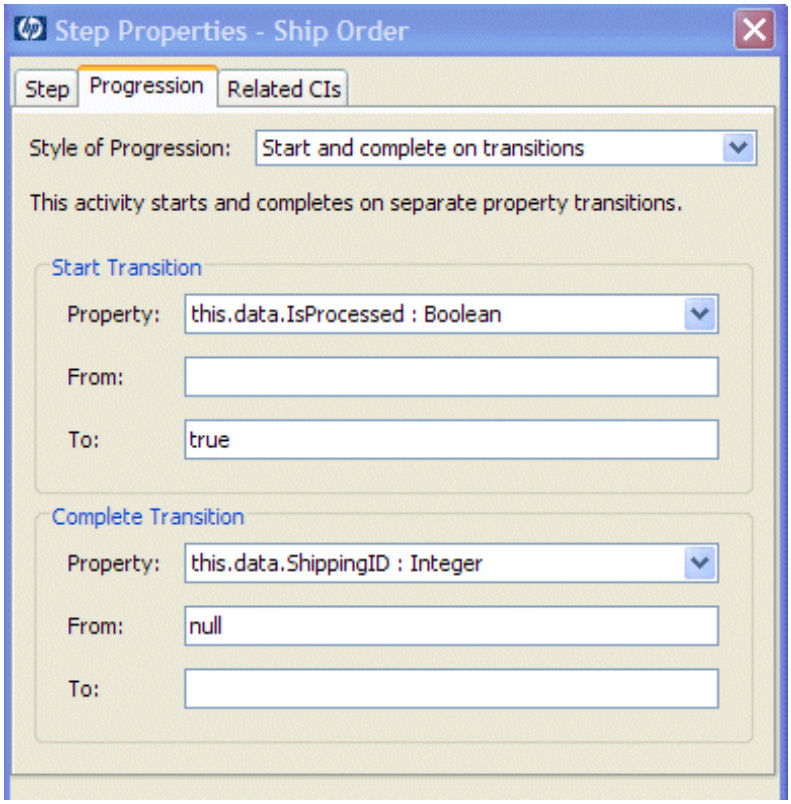
- Configure the `Process Order` Step progression rules as follow:



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`.

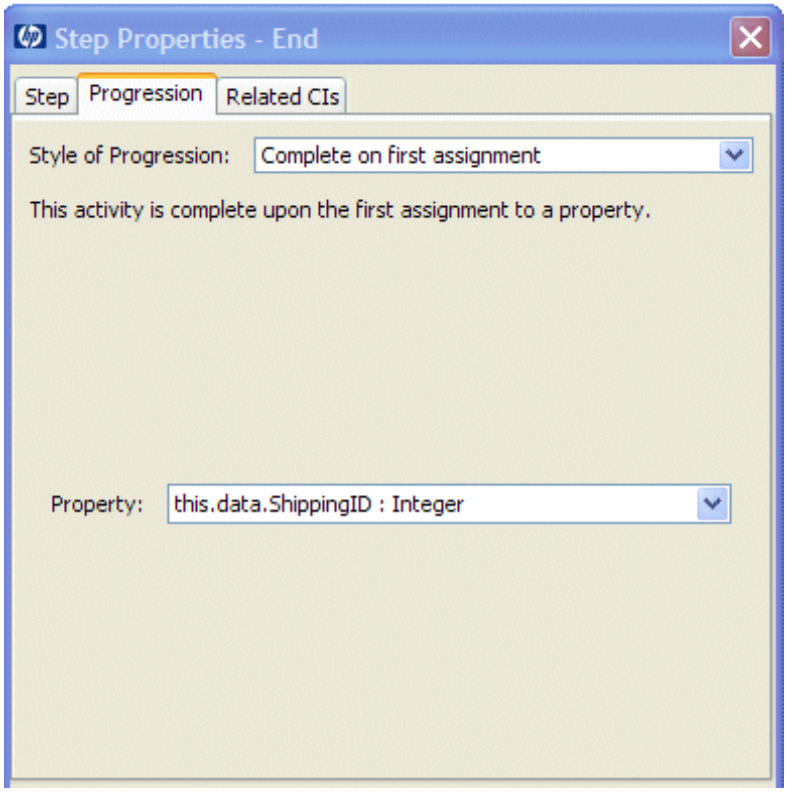
- Configure the `Ship Order` Step progression rules as follow:



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 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 BPI 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 *BPI 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`

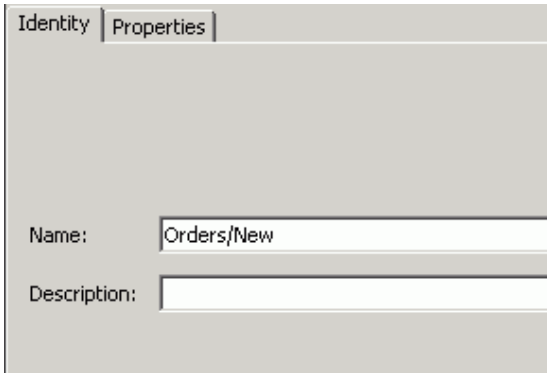
or simply click on the new event icon:

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

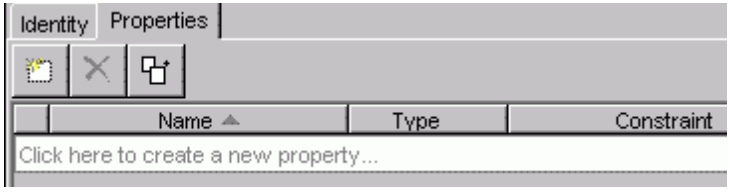
You specify the name of this event definition and (optionally) a description.

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:



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



You can go through and manually create new properties for this event by clicking where it tells you to.

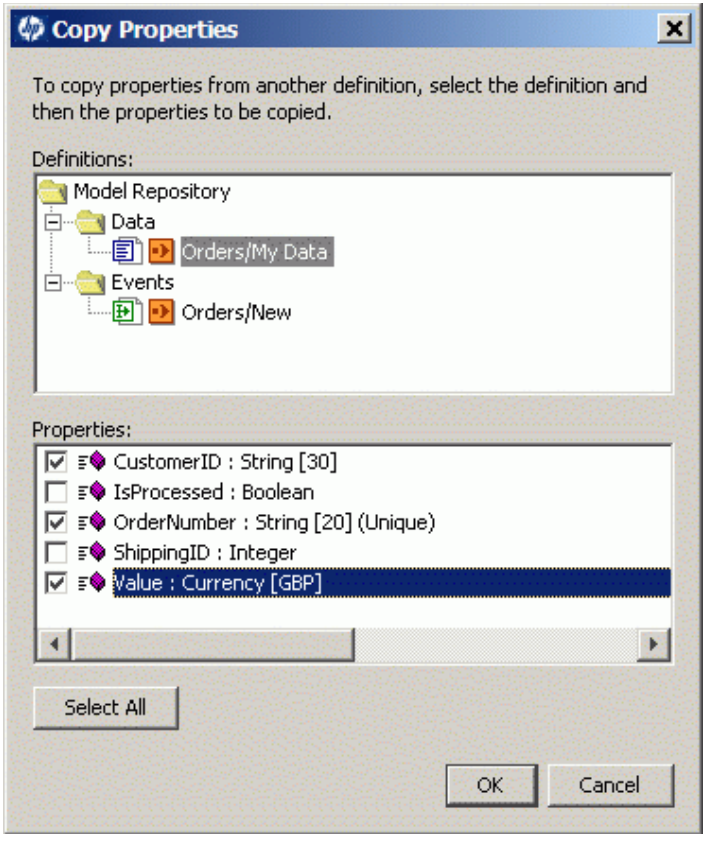
However, by using the `Copy Properties` icon, you can select properties from an existing data definition and the BPI Modeler creates properties to match their name and definition.

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.

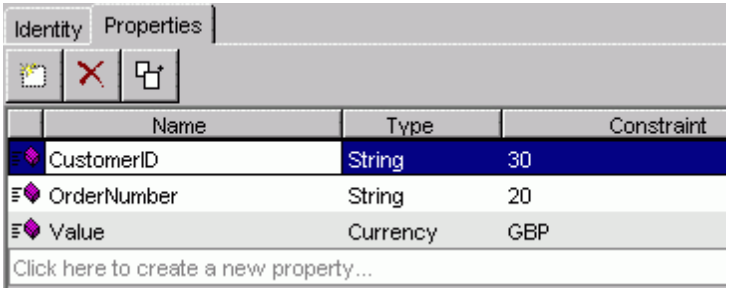
When you click on the copy properties icon, you are presented with a dialog. You then:

- 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:



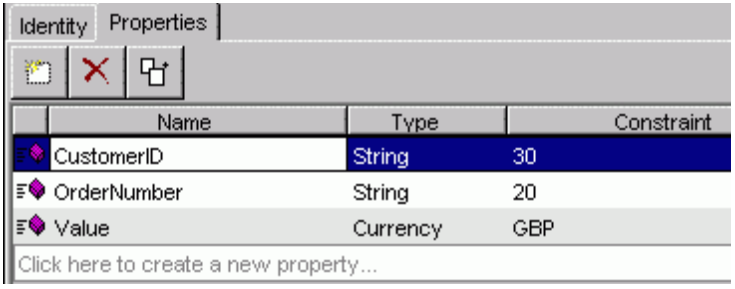
You press OK and it adds these properties to your event definition.


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:



- 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 BPI 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.

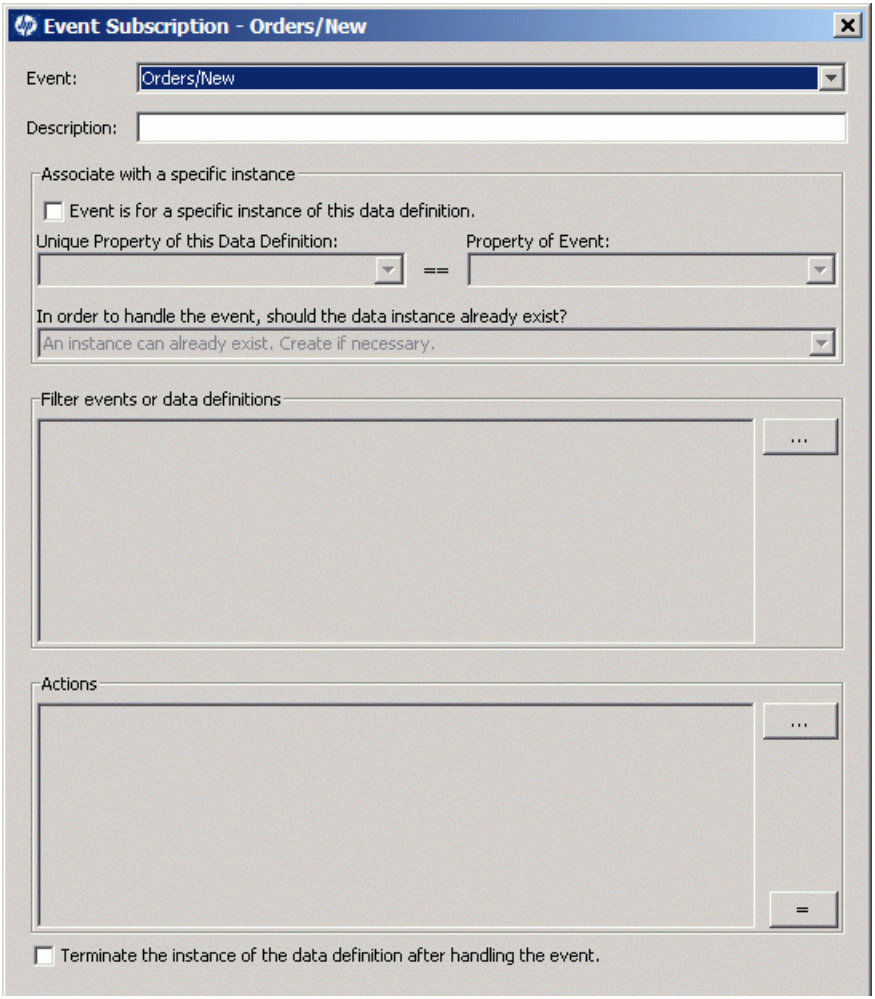
The way you do this is by configuring the data definition to **subscribe** to these events.

Setting Up A Subscription

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

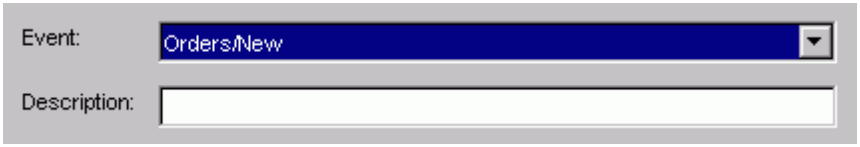
- Double click on the data definition (for example: `Orders/My Data`)
- In the right-hand pane, click on the `Subscriptions` tab
- Click to create a new subscription

An event subscription dialog similar to the following then appears:



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

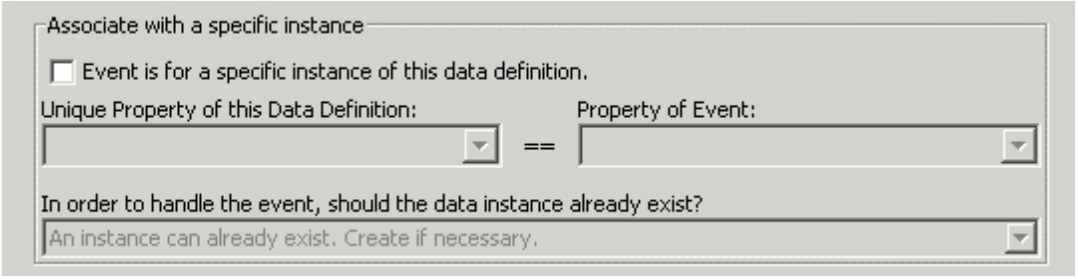
Event and Description



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

You can optionally enter a description.

Specific or Not



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.

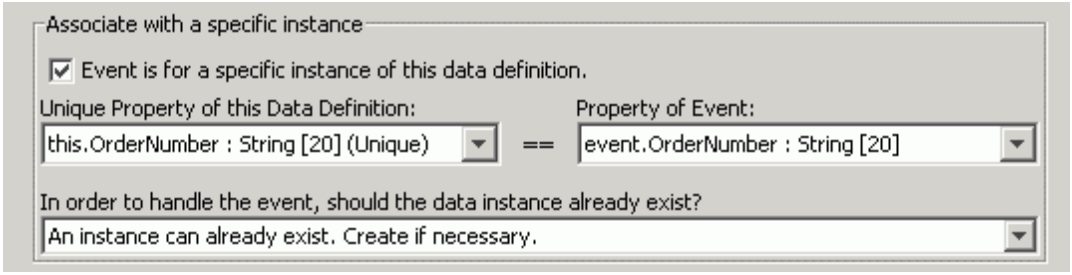
Or, you might not check the box and thus this event affects **all** instances of your data definition. This allows an event to update all the data instances with information.

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.

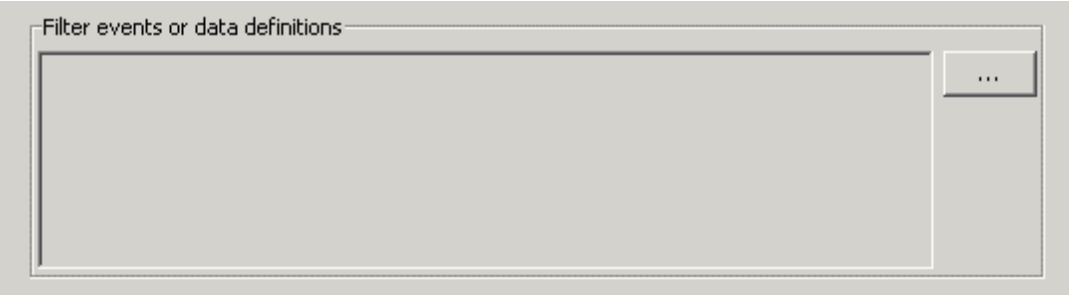
For example:



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`.

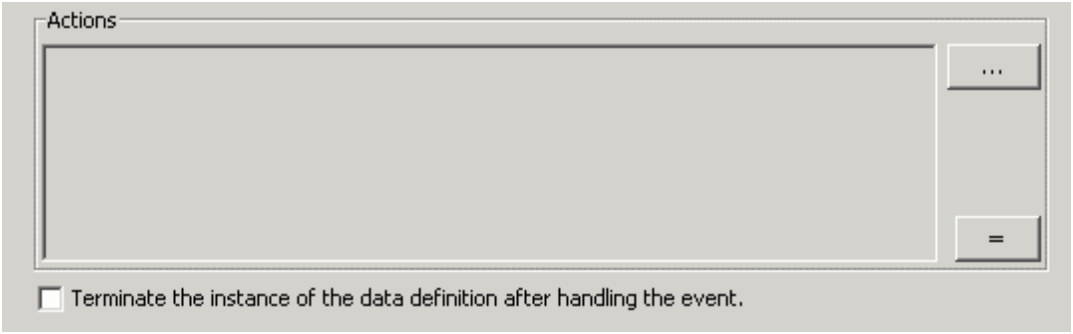
Example 2:

```
   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:

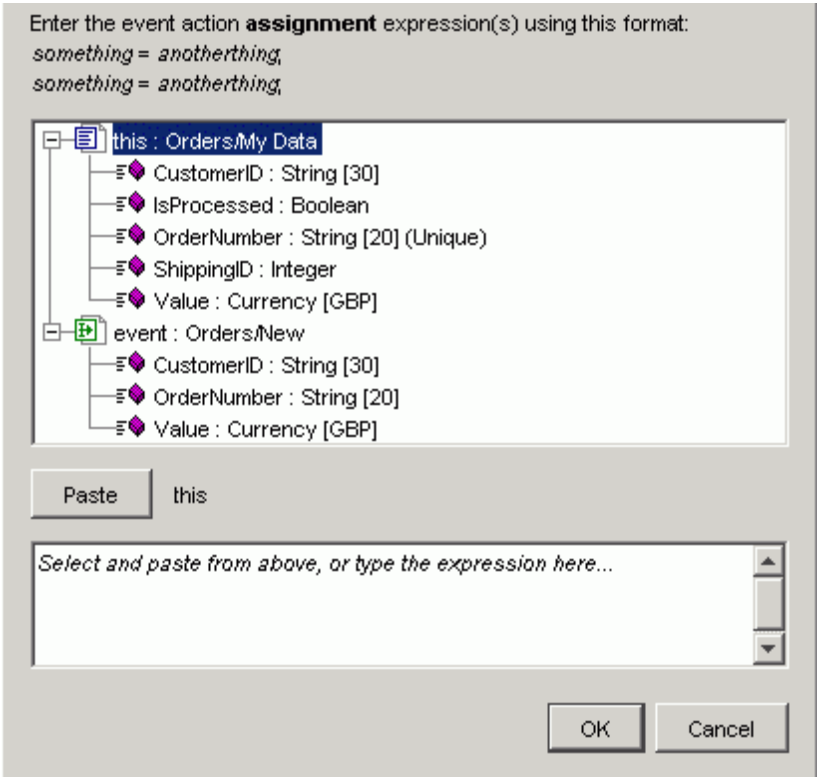
```
this.Property1 = event.EventPropertyA;
this.Property2 = event.EventPropertyB;
...
```

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 yoursel, or:
  - Select the data property
  - Click the `Paste` button
  - Type in =
  - Select the event property

- Click the Paste button
- Type a semicolon `(;)`

- 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 Business Impact Engine is able to handle basic type coercion as described in *Business Process Insight Reference Information*.

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 BPI 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

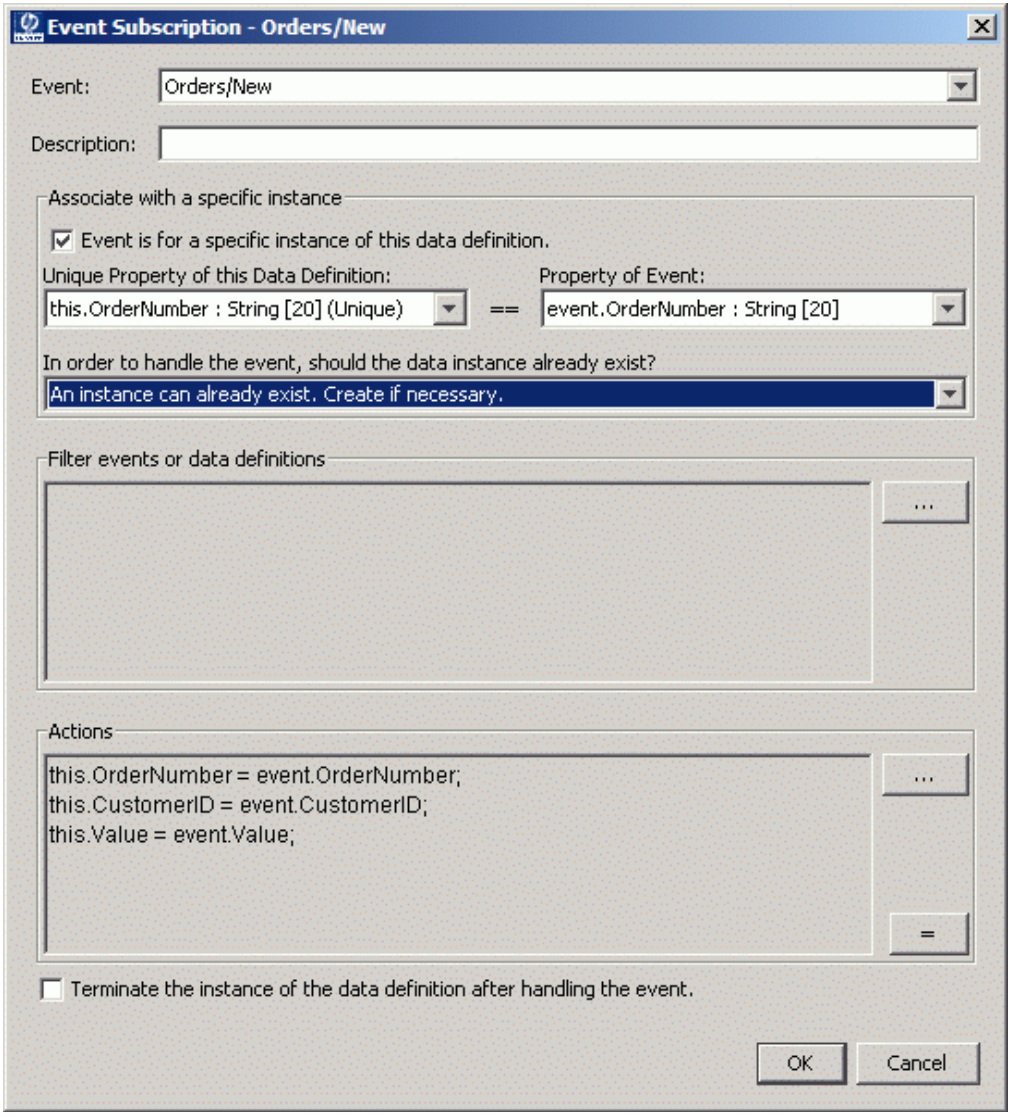
When the subscriptions dialog shows:

- 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 BPI Modeler is able to automatically configure these actions for you.

Your subscription should look something like this:



- 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;`

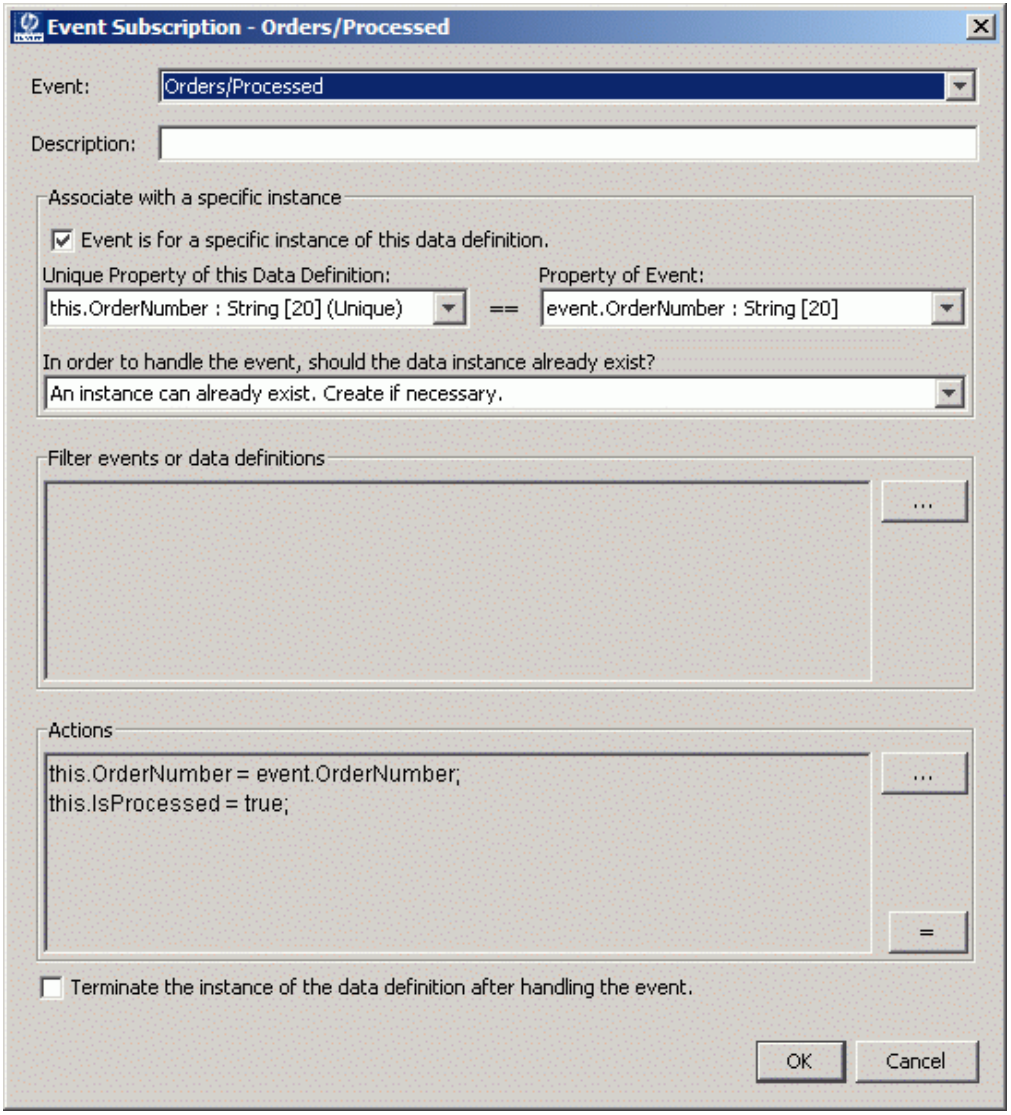
  Append a carriage return after the ";" at the end of the first line. In other words, start a second line.

- 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

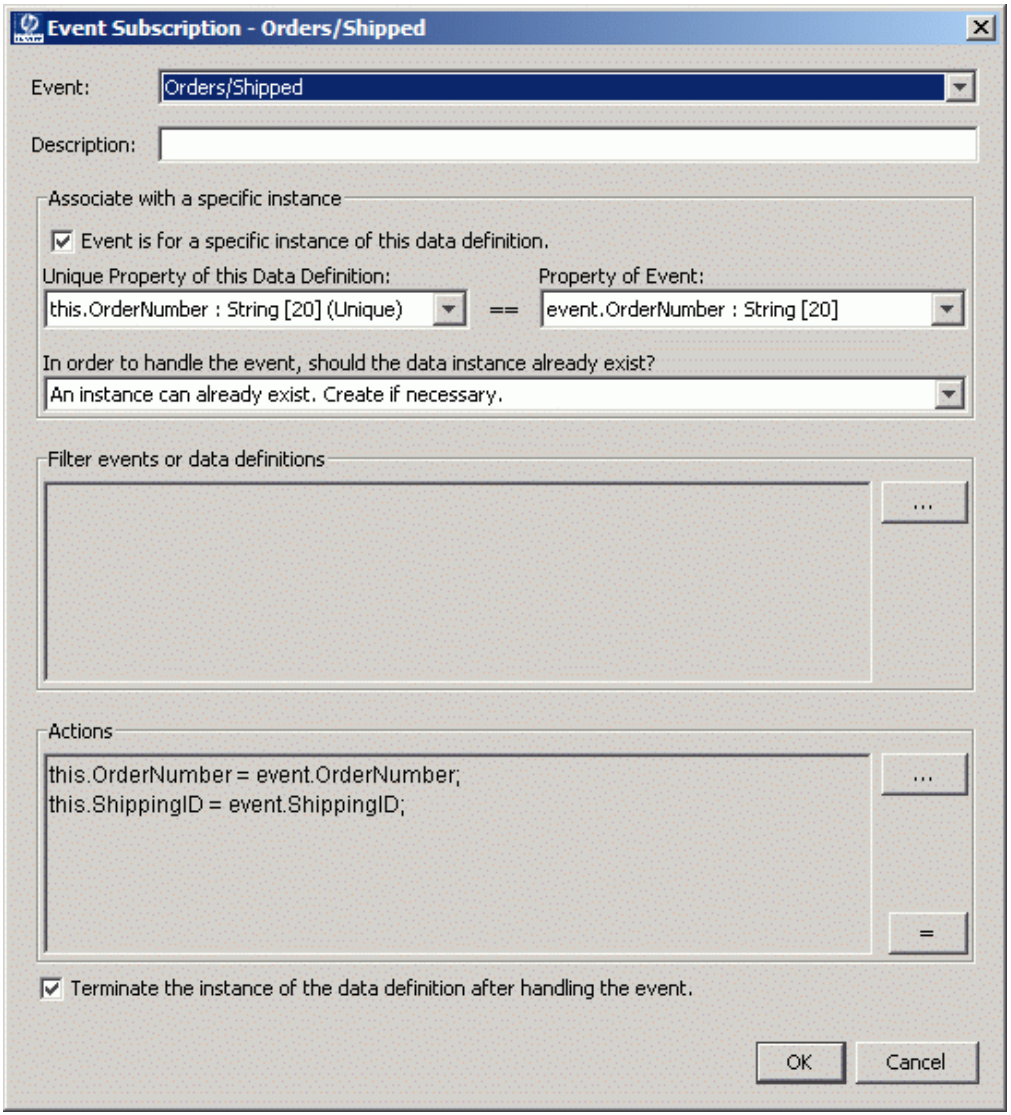
Your subscription should now look something like this:



- 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:



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 BPI 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 Business Impact Engine.

Deploying the Process

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

You deploy a process from within the BPI Modeler.

You simply:

- Select the process name in the Navigator pane
- Then either
  - Click the green deploy icon 
  - Or select: **File->Deploy...**

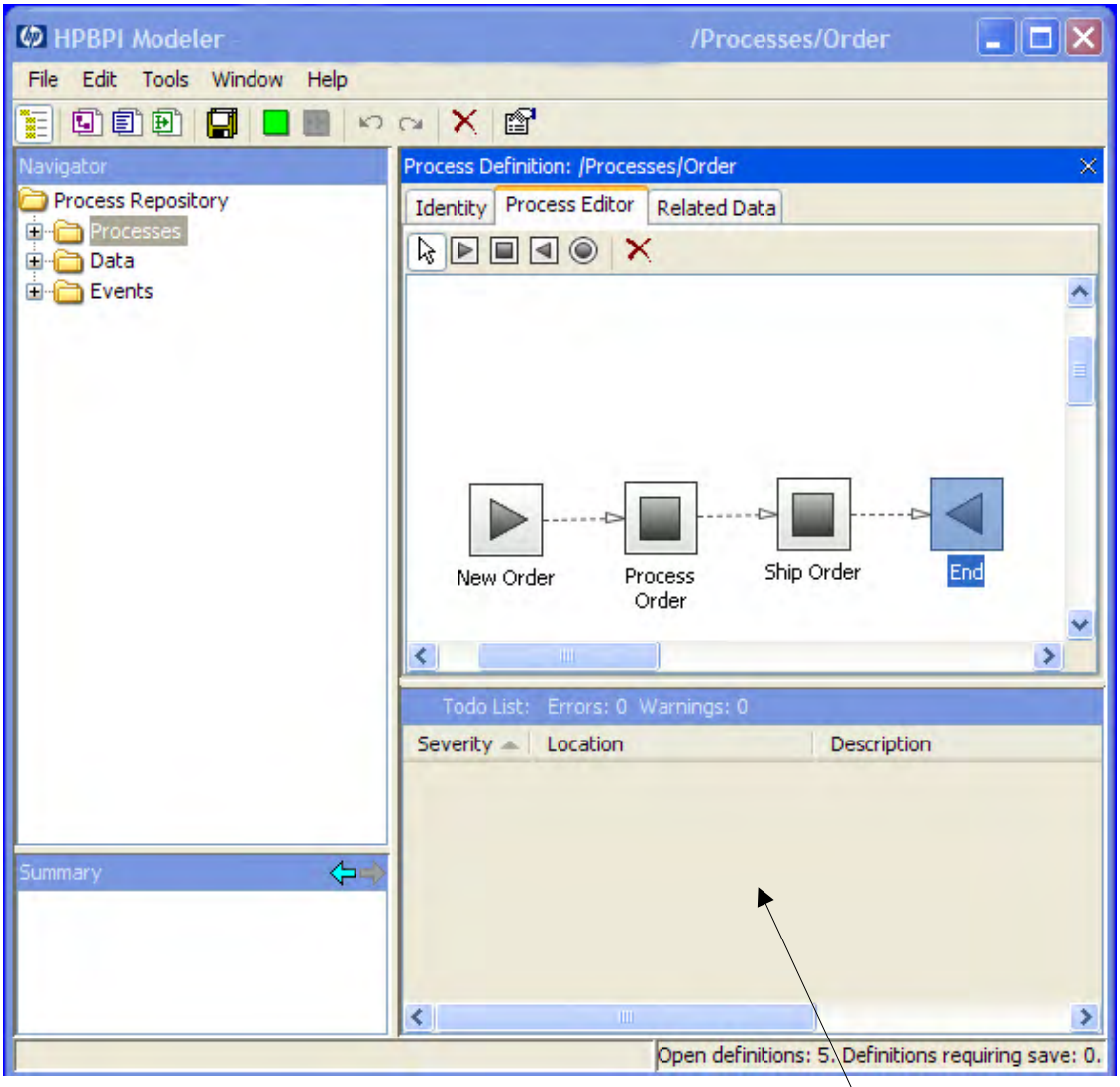
  This then deploys the selected process and any associated definitions that have changed since the previous deploy (if any).
- You see a pop-up dialog listing the definitions that are to be deployed and you can then click OK (or cancel)

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 BPI 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 BPI, how can I see it?"

To view the deployed process you the BPI Application Health pages, which enable you to view any process and see basic information about it.

To access the BPI Application Health pages:

1. Select:

   `Applications > Business Process Insight`

2. Click the `Health` tab.

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 BPI process definition. For more details on how to configure the actual events from the underlying application systems you can refer to the *BPI 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 BPI. 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 *`BPI-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...

Deploy the Process

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

  You should see that it deploys:

  — The three data events

  — The data definition

  — The orders process

View the Deployed Process

1. Select:

   `Applications > Business Process Insight`
2. Select the `Health` tab.
3. Select the `Orders` process from the `Selected Business Process` drop down list.

   You should then see the `Orders` process diagram.

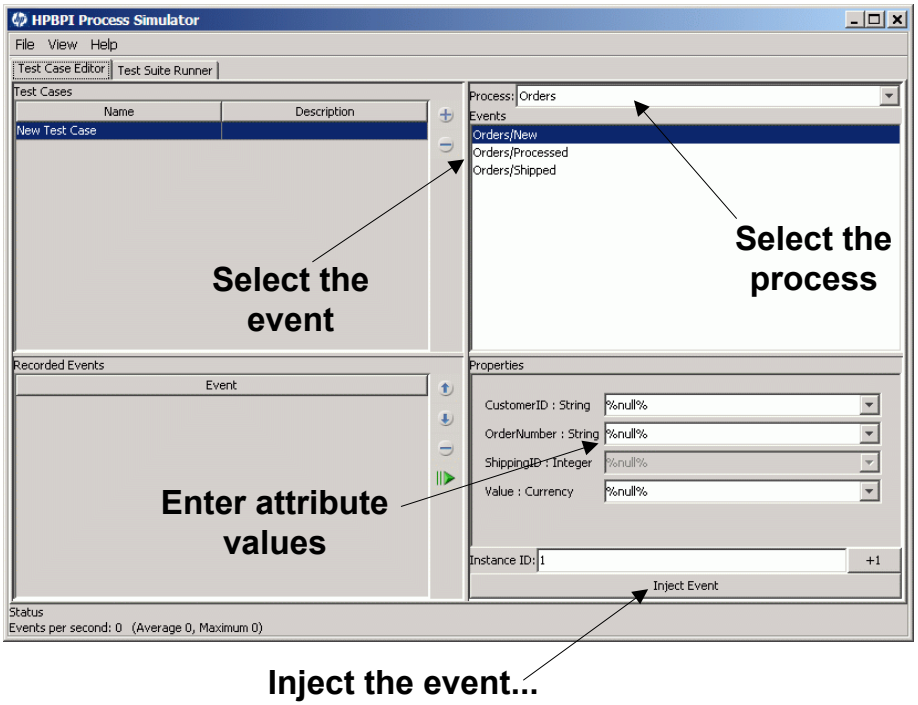
   The Summary details page also shows that there are currently no active instances.

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 BPI Server
- cd to the directory *`BPI-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.

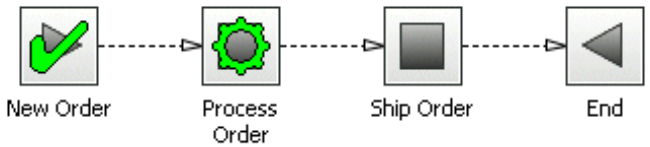
Back in the BPI Application

From the Health page you can now:

- See that you have a single instance of the `Orders` with a status of Healthy.

  The Process Diagram shows a summary of all the instances of the Orders process, including the number of active instances at each Step and the total value of the Orders process at each Step.

- Click the `Healthy` link on the `Summary` tab to display the `Process Instances` dialog.
- Select `Ord1` from the list under `Identifier`.
- The process diagram for this instance is displayed as follows:



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

The blue circle with the question mark is related to the status of the IT operational resources, and as there are not any defined, their status is shown as `Unknown`.

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 Health page and you should see the process instance has moved on to the `Ship Order` Step.

  On the process instance page, click the Data Definition tab and 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.

- Click the Completed link on the Summary tab.

- Select `Ord1` from the list under `Identifier`.

  All Steps in the Ord1 process now show as completed.

- 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

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

Indeed, the integration between the BPI Server and Business Availability Center provides more than just the connection of your IT operational resource 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 the BPI Server with Business Availability Center.

Configuring the Connection to BAC

Refer to `Using Business Process Insight` for full details of how to configure your BPI system to connect to BAC. The following is an outline of the steps that you need to complete:

1. Configure the connection from BPI to BAC so BPI can send data samples to the appropriate Gateway Server (or Load Balancer).

   In most cases this is done when you install the BPI Server. If it is not, you need to configure the settings using the Administration Console.

2. Configure the connection from BAC to BPI, so BAC knows where the BPI Server and the BPI Instance database are located and can sent the BPI Server status updates relating to the IT operational resource CIs.

   This is done through the BAC Infrastructure settings.

When you have successfully completed the configuration, the BPI Server sends data samples to BAC and receives operational resource status information from BAC.

Some of the settings for these options can be configured from the Administration Console, HP Business Availability Center settings as described in *Using Business Process Insight*.

Data Samples

You need only modify these if you want to change the properties relating to the frequency at which data samples are sent to the BAC Gateway Server. The BPI Server sends the following data samples to the BAC system:

- The current active count and active value for each deployed process.
- The current active count and active value for each deployed process step.
- The current business health of each deployed process.
- Values for any BPI Monitors instance thresholds that you have defined for your deployed process. Refer to the *BPI Integration Training Guide - Defining Business Process Monitors* for details of how to define Business Process Monitors.

Operational Resources

Within BAC all objects are defined as CI types within the UCMDB.

You need to set up the appropriate IT Infrastructure CIs and any dependencies between these CIs and other CIs within the UCMDB. For example, you could configure the state of a Business Process Step CI to be dependent upon the state of a particular database or application system.

The `CI Poller settings` provide options for the link to Business Availability Center IT hierarchy (CIs).

BPI polls Business Availability Center (as configured by the `Status event poll interval`) and retrieves the status details for all CIs within the view specified as `View name`. The BPI Server then pulls out the status for each CI and sets this as the status for the corresponding Step with BPI.

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, the BPI Server 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 BPI processes deployed to Business Availability Center appear within the `Business Processes` view.

Business Process CIs

The BPI Business Process data is stored in the UCMDB as configuration item (CI) of several types.

Let's now inject some events into your `Orders` process and see what data is shown in the BPI Application, and the corresponding CIs shown in 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 the BPI Server 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 the BPI Server.

Open the BPI Application Health Page

1. Open the Health page for the BPI Application:

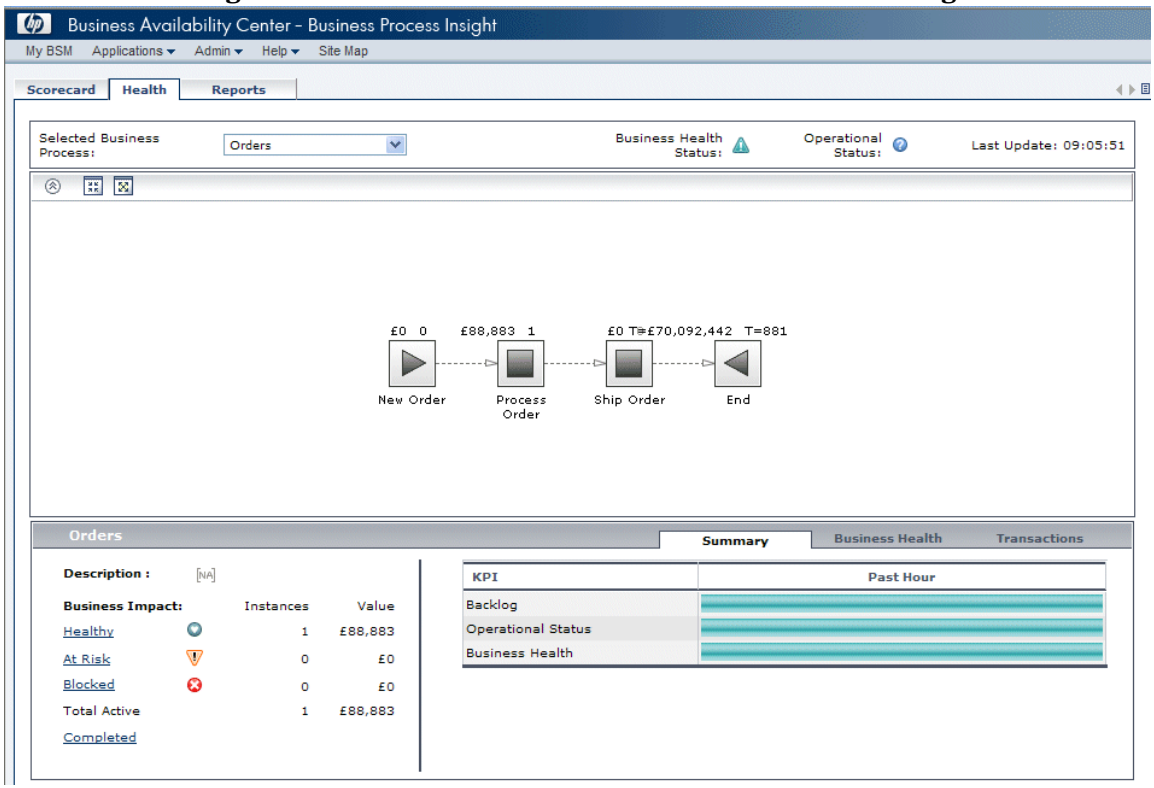
   `Application > Business Process Insight > Health`
2. Select the `Orders` process

   You should see the numbers above the process Steps changing, indicating that instances are being processed.

   You also see the overall Business Health Status and Operational Status at the top of the page.

For example, your Dashboard results for the Orders process might look as shown in Figure 8.

Figure 8 BAC Dashboard - Orders Process Health Page



where:

- The Backlog is the “active value” for the Step or process.
- In this example the `Backlog` bar shows that all current process instances are healthy.

  If some instance were blocked or at-risk, then this `Backlog` bar would provide a visual representation of the relative sizes.
- Business Impact information is broken down into the number of instances for the process that are Healthy, At Risk and Blocked. You can also select a Step in the process to have the same data presented, but for the Step and not the whole process.

- You can drill down to the details of the individual instances, by selecting Healthy, At Risk, Blocked or Completed.
- You can select the Business Health Tab to get information relating the Business Objectives defined for the process.

  You set objectives for `Business Process` KPIs from:

  `Admin > Dashboard > KPIs`

Business Health When There Are No Active Instances

The BPI Server sends data sample values to the BAC Gateway Server as often as is configured within the BPI 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 it's Steps) at the exact moment that the BPI Server sends a data sample, then the Business Impact within Business Availability Center will show a status of `Unknown` (blue circle with a question mark) as shown in Figure 8.

CI Poller

These settings enable the BPI Server to receive status information, which relates to operational services that you have associated with BPI Step CIs.

Adding Dependencies

Within BAC 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 an application system. The way you link up your CI dependencies is up to you.

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.

IT Operational Resource Status Change in BAC

Suppose the status of a dependent CI for the `Ship_Order` CI changes to *Critical*.

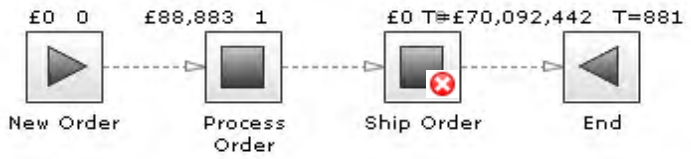
When this occurs, the Business Impact shows a “yellow and red bar” that indicates that you now have some process instances at-risk (yellow) and some process instances blocked (red). The relative sizes of the colors in this bar indicate the relative percentage of at-risk and blocked. If there were some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

Operational Resource Status Change in BPI

When a Business Process Step CI changes its status within Business Availability Center, this status is reflected back within BPI. Remember that the BPI Server 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 at the BPI Server, and you can see this in the BPI Application. The `Orders` process might look as shown in Figure 9 on page 105.

Figure 9 BPI Dashboard - Ship Order Step Critical



Synchronization Matters

Dashboard displays the Business Health of a process. The data that determines the coloring of how this Business Impact bar is displayed, is sent from the BPI Server. This data is sent from the BPI Server according to how you configure the `Data samples settings` within the BPI Administration Console.

Dashboard also shows the status of your CIs within the UCMDB, and these statuses are displayed as soon as they change within the UCMDB.

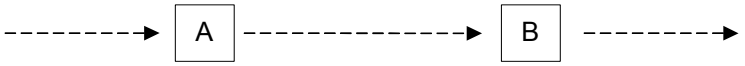
So it is quite possible that when you view your process data within Dashboard that you might see a CI status change, but not see the Business Impact bar change to reflect this until some time afterwards.

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 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 BPI 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:

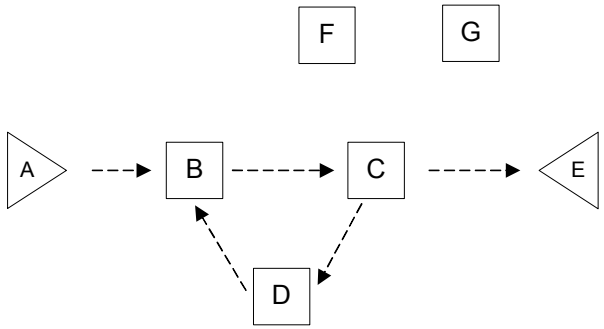
This tells the Business Impact Engine (at run time) to trap any situation where Step `B` is started before Step `A`. If Step `B` does start before Step `A` then the BPI Server issues a `Process Out Of Sequence` notification. (The destination of this notification can be configured within the BPI Notification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

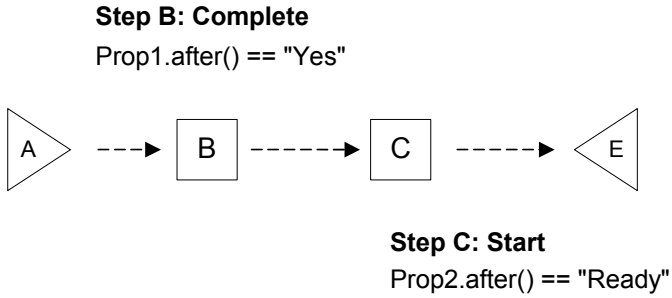
Active Processes With No Active Step(s)

It can happen that you deploy a process and then, when monitoring it, 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 the BPI Server 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
  ```

  This 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 Business Impact Engine has an active process instance, which shows in the BPI Application; however. the BPI Application 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. the BPI Server quite happily monitors your process and reports what it can. It just might surprise you when/if it happens.

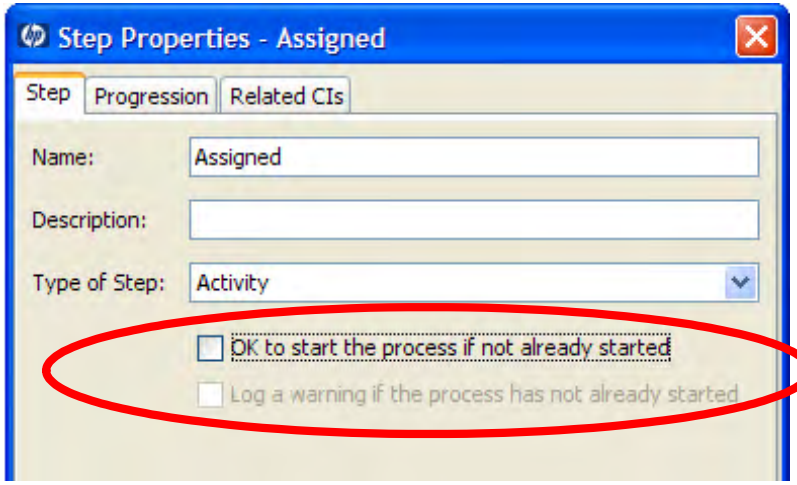
If you do have this situation where you have active process instances that are not showing up in your Step instance list, then it might mean that you need to model more of your Business Process, or it might mean that you have made an error in one or more of your progression rules, or it might be by design and 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:



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 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 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 Business Impact Engine's log file, which can be viewed using the BPI 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

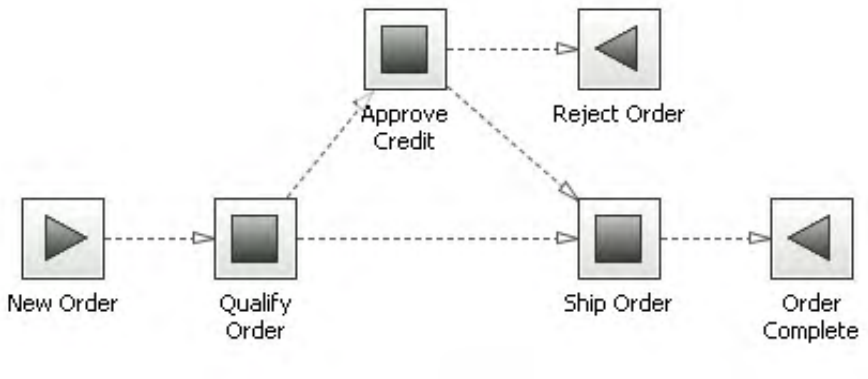
A Junction Step can be placed anywhere within a process diagram. You can have more than one Junction Step within a process diagram. You may choose to have no Junction Steps within your process. Junction Steps are optional.

So why does BPI 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 10:

Figure 10 Order Process Diagram



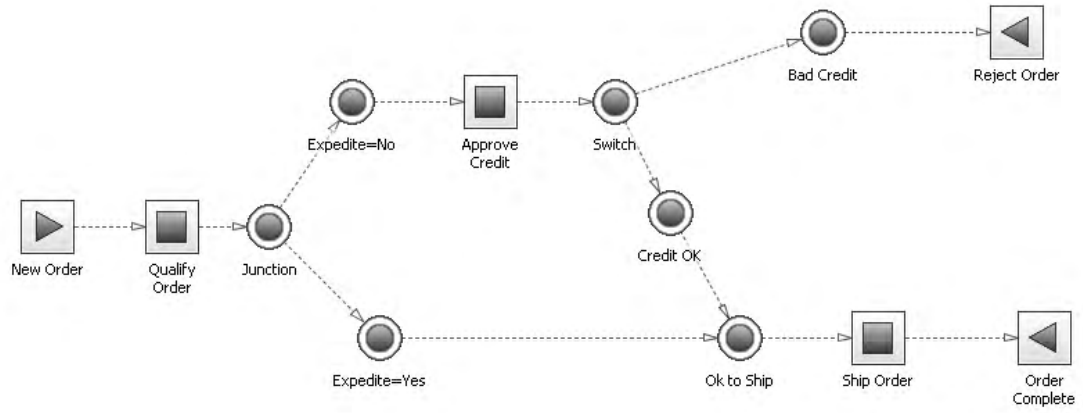
The above process shows that when an order comes in, you first qualify the order to see if the customer is a special customer or not. In the case of a special customer's order, you choose to expedite this order as there is no need to approve their credit. You can go straight ahead and ship the order. 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 10 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 11:

Figure 11 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 Process Repository

When you are running the BPI Modeler you are connected to the Process Repository component that manages the data between the Modeler and the Process Repository database.

Process Repository Startup

The Process Repository is started and stopped using the `Process Repository` option within the BPI Administration Console.

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

When the Process 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 Business Impact Engine. Hence it is always a good idea to start the Business Impact Engine component before starting the Process Repository.

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

Thus, if you restart the Process Repository and then re-run the BPI Modeler you see a clean indication of the definitions that are actually deployed to the Business Impact Engine.

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

Importing And Exporting Definitions

The BPI Modeler allows you to import and export definitions.

The most obvious example is where you have defined a process - and its associated Data and Event definitions - 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 BPI 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 content of the Process Repository database in one go, using the BPI Repository Explorer - see Exporting Definitions on page 119 for more details.

Importing

To import definitions into the BPI 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 BPI Modeler as being **undeployed**. It is then up to you to deploy this definition to the Business Impact Engine.

Process Repository Explorer

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

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

Full details of the featured of the Process Repository Explorer can be found in *Using Business Process Insight*.

Running the Process Repository Explorer

You access the Process Repository Explorer, as follows:

`Admin > Business Process Insight > Process Repository Explorer`

View Details

The Process 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.

Recycle Bin

When you delete definitions within the BPI 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 BPI Modeler.

To see the definitions currently marked as deleted, you use the BPI 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 BPI Repository Explorer
- Then select the `Cleanup Recycled` option

Restore

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

Exporting Definitions

The Process 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 BPI Process Repository database-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 BPI Modeler deploys the selected definition and any dependencies. This is why, when you deploy a process, the BPI Modeler deploys all the parts necessary to enable that process to be deployed and run.

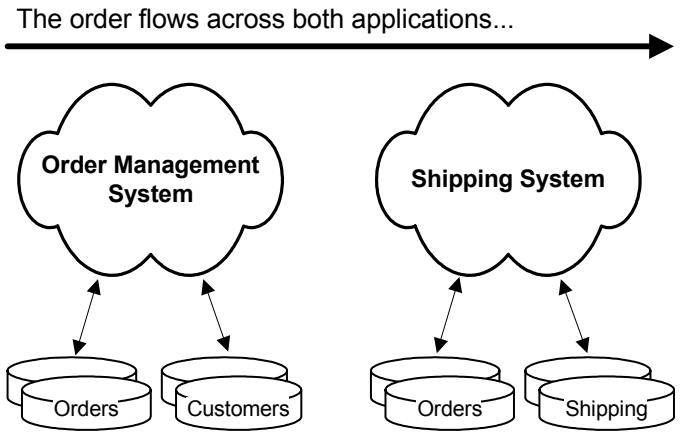
There is nothing stopping you deploying a single Event definition. in which case, only that event definition is deployed. Similarly, you can deploy just a Data definition and the BPI Modeler deploys that Data definition and any, as yet undeployed, Event definitions that this data definition is subscribed to.

But don't forget that the BPI 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 BPI 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 BPI Modeler deploys only 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.

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 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 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 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 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 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 Business Impact Engine in time, the 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 125

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 BPI 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:

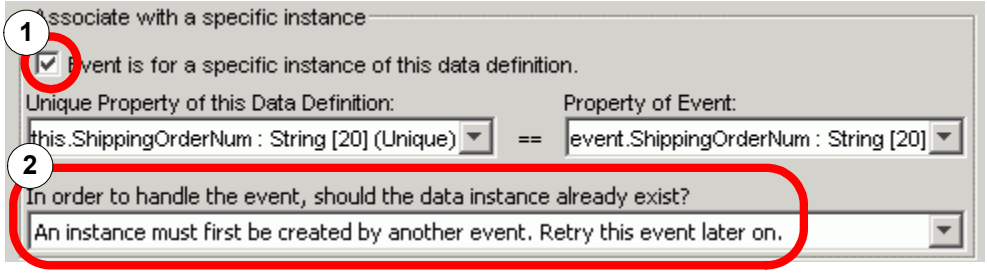
```
An instance can already exist. Create if necessary.
```

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 12 Retry the Event



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 Business Impact Engine marks this event to be re-submitted, and hands this event back to the BPI 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 Business Impact Engine processes them in the correct order.

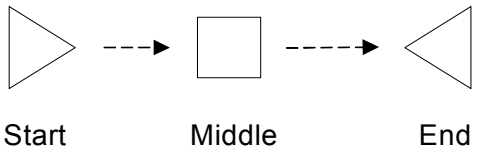
Lab - The Process Seems To Stop Working

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**
Key_Field	String	10	Yes
Value_Field	Integer		

- 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`.

- 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

- View the process within the BPI Application:

  `Applications > Business Process Insight > Health`

  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
```

- You should hopefully be able to see this new process instance and see that it is indeed sitting in the `Middle` Step.

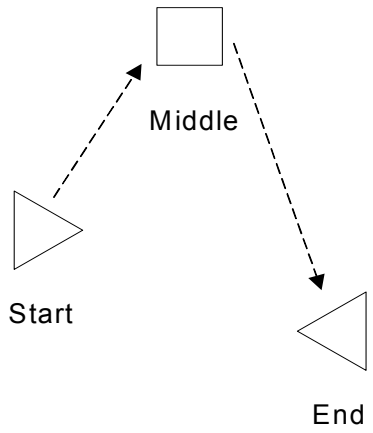
Wonderful! Your process is working.

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 BPI Modeler and make some changes...

- In the BPI 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...

- View the process using the BPI Application as follows:

  `Applications > Business Process Insight > Health`

  You should now see that `Simple` is listed in both the active 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 check within the BPI Application 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 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 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 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.

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 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 *Business Process Insight Reference Information*.

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.

  Alternatively, you could also use the `onEvent` method:

  ```
  onEvent("Order Created")
  ```

  This starts the Step when an event named Order Created comes in that creates an instance of the related Data definition; see section Event-Driven Progression Rules on page 140 for more details.

- **`.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 BPI 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.

Event-Driven Progression Rules

With the progression rules described in the previous section, you need to create a Data definition, and then configure an event subscription for the Data definition. Progression rules for the Business Process are then defined in terms of changes to the Data definition.

If the arrival of a particular event is all that is needed to progress a Business Process, then this information can be obtained directly from the business event. This means there is no need to define Data properties where they are not required.

You can define an `Advanced Condition` style of progression rule that enables you to progress your Business Process Steps either based on the arrival of a particular Business Event, or the value of a property within the event as in the following examples:

Example 1:

```
Start Condition:
       onEvent("Start Credit Check");

Complete Condition:
          onEvent ("Credit Check Done");
```

Specifies that:

- The Step starts when the Event definition `Start Credit Check` is processed.
- This Step completes when the Event definition `Credit Check Done` is processed.

Example 2:

```
onEvent(“Order Updated”) && event.Status == “Credit Check Done”;
```

This is an example of a Start Condition for a Process Step based on an event property. The expression shows a process Step is to be started when the business event `Order Updated` is received and the business events contains a `Status` attribute of `Credit Check Done`.

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.

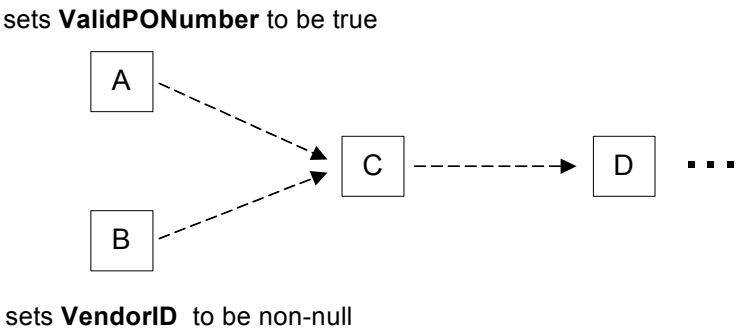
So be careful. Remember that **all** Step progression rules are evaluated for each event that affects the process's related data instance.

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:



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:

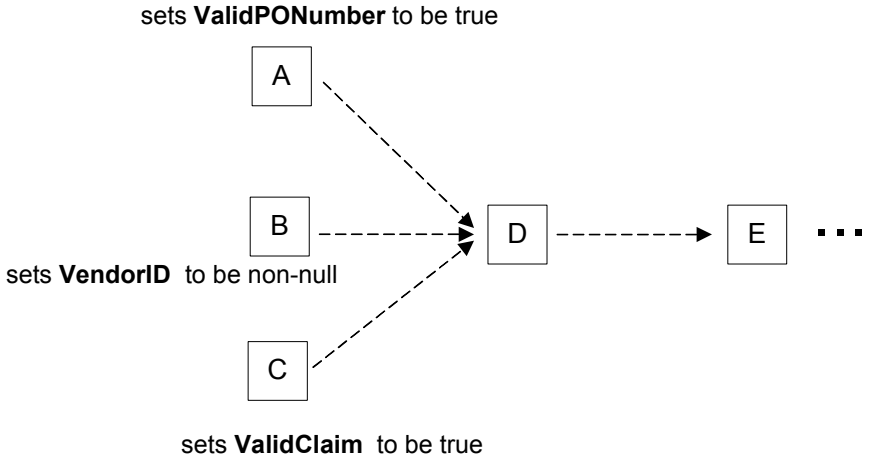
- That at least one (or both) of the attributes `ValidPONumber` and `VendorID` have just been changed by the event.
- And the attributes have the set of values that you are testing for:

  ```
      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.

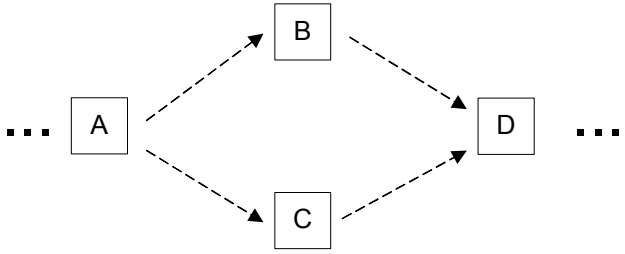
Out of Sequence Business Data Events

Let's look at how to write progression rules that cope when business data events arrive at the BPI Server out of sequence.

Consider the following example:

- The process diagram is as shown in Figure 13.

Figure 13 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 13 on page 147. 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 the BPI Server 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 the BPI Server 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 *BPI Integration Training Guide - Business Events*.

However, if the events arrive at the BPI Server out of sequence, then you will have problems.

For example, if the events were to arrive at the BPI Server 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 13 on page 147, 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 13 on page 147) 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, the BPI Server does not maintain a time-sequenced set of values for an instance of a data definition. That is, the BPI Server only keeps the most recent value for a data attribute. Thus, when event `3` comes in setting `MyStatus` to `B`, the BPI Server 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`.

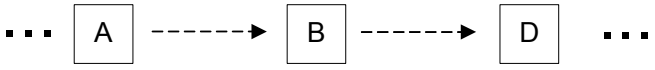
Interpreting the Complete Condition

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 14.

Figure 14 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, the BPI Server is able to cope with this and the process instance is recorded correctly.

For example, suppose the events arrive in the following order:

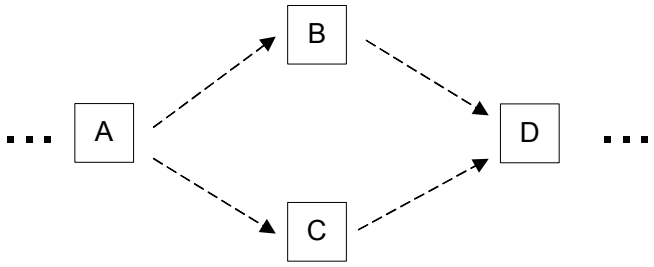
```
Event 1: MyStatus=A, BPI_GeneratedDate=10:01
Event 2: MyStatus=D, BPI_GeneratedDate=10:03
Event 3: MyStatus=B, BPI_GeneratedDate=10:02
```

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, BPI 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 15.

Figure 15 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.

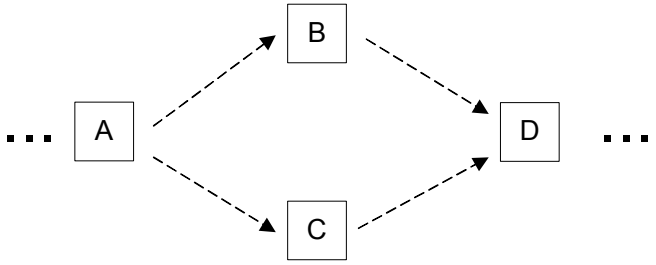
A Specific Event, or Status, for Start and Stop Conditions

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 16 on page 154, uses the data attribute `MyStatus` to progress the process.

Figure 16 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 the BPI Server 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 the BPI Server 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 to the BPI Server, and thus guarantee the events are sent to the BPI Server 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.

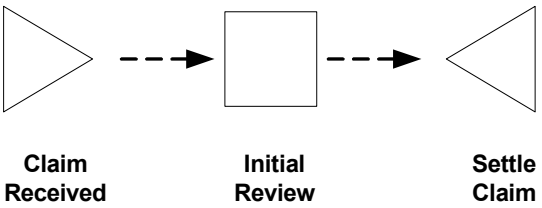
Initial Process Definition

The first task is always to gain an understanding of the underlying Business Process and then to sketch out the high-level BPI Business Process that you want to use for monitoring.

This initial BPI 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 17 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

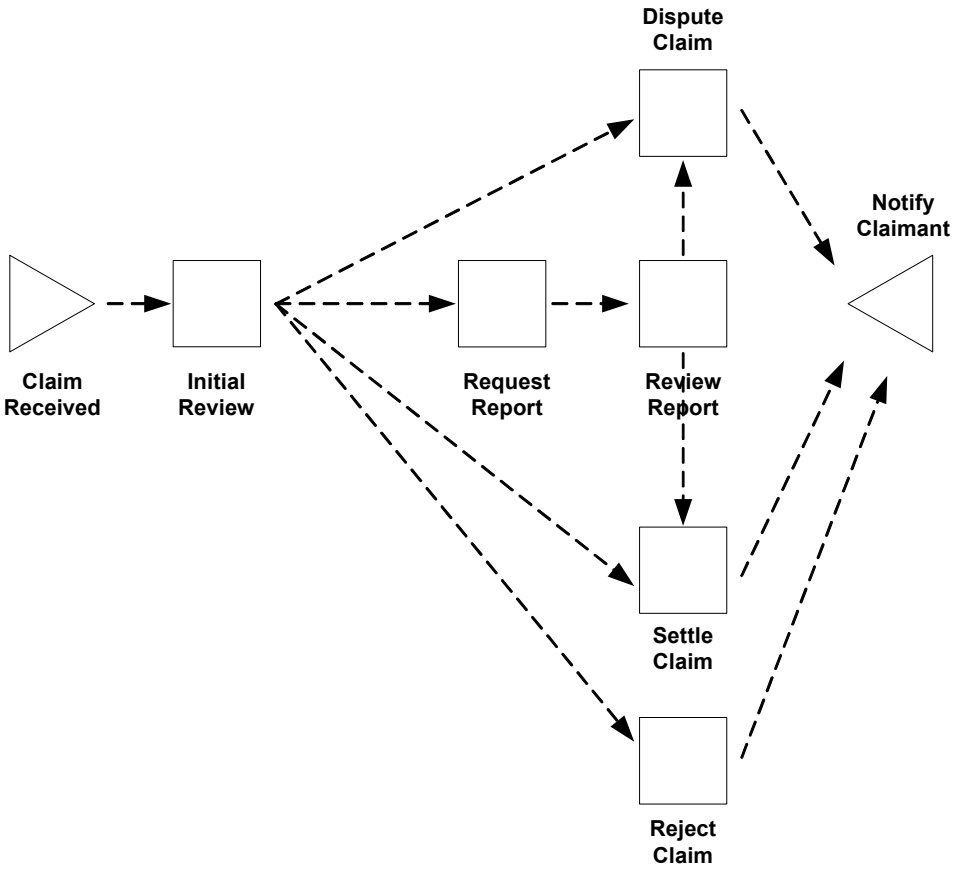
You decide that you need to expand the process definition to include more detail...

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 18 Stage 2 High-Level Insurance Process



where:

- **Claim Received**

  A process instance is initiated when the 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 though 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 BPI Modeler.

Lab - Defining the Basic Process Diagram

In this lab you defines the process diagram.

The Process Diagram

- Run the BPI Modeler
- Create a new process, called: `Insurance Claim`
- Draw the process so that it matches the process diagram shown in Figure 18 on page 161

  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.

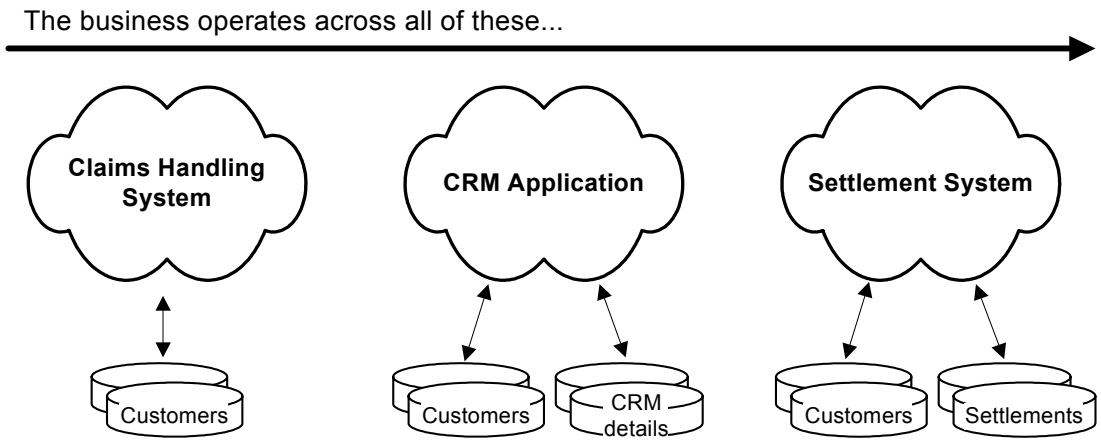
Be aware that if you define too much data, the performance and manageability of the process are impacted, so you need to make sure that you include only the data that you need; you do not need to include all the data that is available. Too much data also creates the integrator problems when trying to 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 BPI 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 BPI 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:



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 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 BPI 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 122.)

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, value of the claim.

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 BPI 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 *BPI 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 BPI 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 BPI Application 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 BPI Application:
  - **Claims/New**

    ```
    Claim_ID:     10
    Claim_Value:  22000
    ```

    A new process instance should appear in the BPI Application
  - **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
```

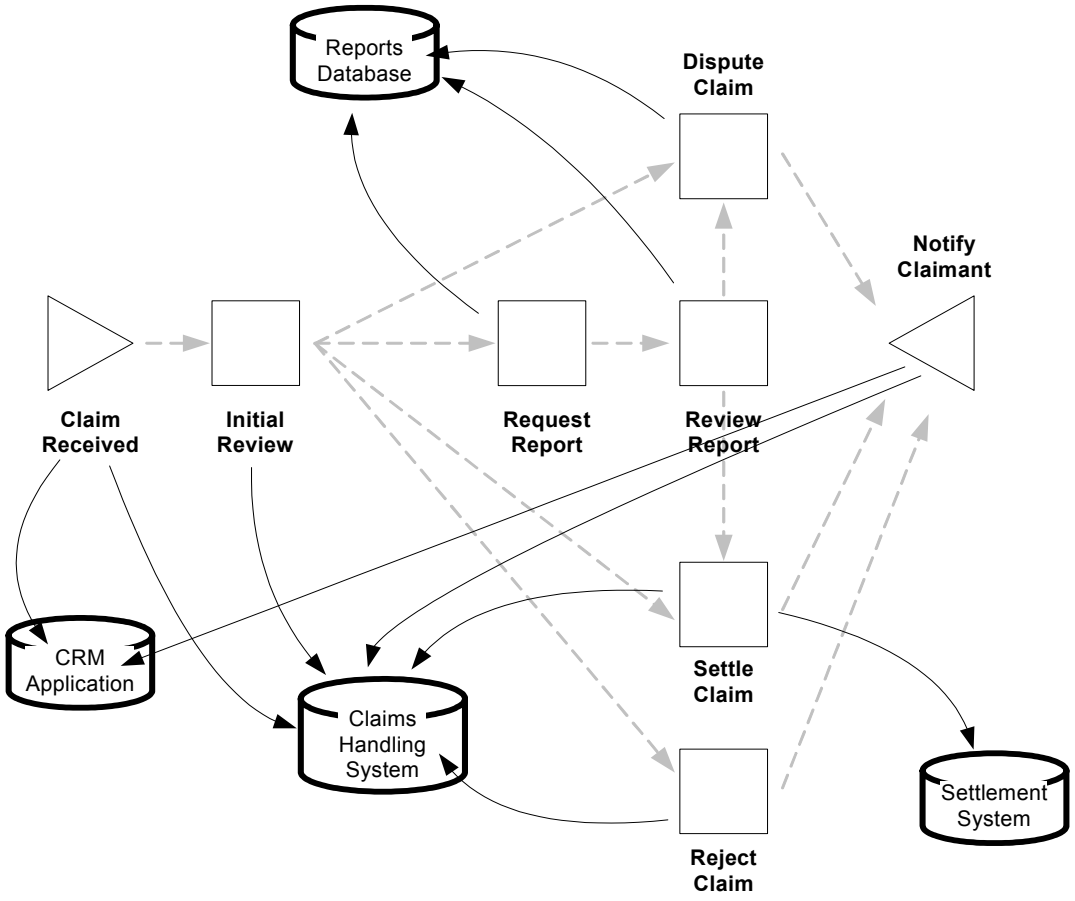
Well done! You have reached the end of the lab.

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 operational resources.

For example, your insurance claim process might have the following IT dependencies:

Figure 19 Insurance Process and IT operational resource Dependencies



Lab - Defining IT Operational Resources Dependencies

This lab assumes that you have access to a Business Availability Center installation.

Enabling the Integration with BAC

Run the BPI Administration Console and enable the integration between BPI and your Business Availability Center installation.

Make sure you enable `CI Poller 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 it's dependent Steps.
- Set up either a new KPI, or an IT dependency, for at last 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 BPI Application.

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