BPL

The Business Processing Language – for orchestration and long-running business processes.

Ensemble can orchestrate calls to external systems. Very often this is done to implement or automate a long-running business process – that is, real business processes where people interact with a series of different systems to complete complex tasks. BPL (the Business Processing Language) provides a graphical way to create these orchestrations.

General Questions

Q: Is BPL a product?

A: No, BPL is not a product. BPL stands for Business Process Language and it is an integral part of Ensemble. It is the XML 'language' that underlies the graphical business process editor.

Q: Is BPL a graphical tool or is it XML?

Strictly speaking, BPL is the XML, but because it is nearly always viewed and edited through the graphical form, the graphical display is normally called BPL or a BPL diagram. A particular diagram representing a particular example of BPL is often called 'a BPL' even though purists would say that isn't correct.

Q: Where do I start with BPL?

A: You start by either reading the section headed 'Using the BPL Visual Editor' in 'Developing Ensemble Productions', or you could go straight the Business ProcessBuilder in the Management Portal, go to Ensemble >> Build >> Business Processes.

Q: Isn't BPL just another programming language?

No. You could use the graphical editor to build typical business logic with loops and conditionals but that would be missing the point. It is really meant for orchestrating calls to external resources and other Ensemble components. You need the loops and conditions to do that, but if you are using them mainly to manipulate variables or local data, you should think again.

Q: What do I need to know before using BPL – are there prerequisites such as XML or Caché ObjectScript?

A: You don't need to know about XML, you may never look at the underlying XML. You do need to know a little about the Ensemble concepts of sending requests and responses, and the role of Business Services, Processes, and Operations. A little knowledge of Caché ObjectScript is not essential but will be a great help. Someone in the team needs a basic understanding of Caché objects, but it doesn't have to be the person writing the BPL.

Q: Where can I use BPL?

A: You can only use BPL to implement a business process in a production. It can't be called outside a production. BPL is created in a class. You then add a business process item to a production, specifying this as the class name.

Q: Can I edit BPL from a browser?

A: Yes! Starting with 2012.1, you can edit BPL and DTL from the Management Portal as well as with Studio.

Q: Can I put BPL into source code control?

A: Yes, Studio and the Management Portal use a common source control framework. For more information, see the "Using Studio Source Control Hooks" chapter in Using Caché Studio.

Q: is BPL the same as BPEL? Can I use BPEL?

A: No. BPEL (the Business Process Execution Language) is a standard for orchestrating Web Services. When we first designed Ensemble BPL, we modeled it as closely as we could on BPEL but it had some really important things we just couldn't achieve if we had stuck to the BPEL standard.

Q: Where do I learn more?

A: To read more about BPL, start with the "Business Processes" [http://docs.intersystems.com/ens20102/csp/docbook/DocBook.UI.Page.cls?KEY=EGDV_busproc] chapter of *Developing Ensemble Productions*. For more details about the language elements, see the *Ensemble Business Process Language Reference*. [http://docs.intersystems.com/ens20102/csp/docbook/DocBook.UI.Page.cls?KEY=EBPL]

…start with the "Business Processes" and "Developing BPL Processes" chapter …

Q: Why are some processes referred to as 'long-running business processes'?

Long-running in this context doesn't mean they take a long time to execute, like a big batch job, it means it is waiting indefinitely on some external event. That might be waiting for a person to approve the next step in a workflow, or it might be waiting for notification that some information is not available.

So the business process has to just wait, perhaps for a few seconds, perhaps for a few hours or perhaps for days. It can't just pause in its execution because it would be vulnerable to system restarts and it would be taking up system resources.

Ensemble copes with this by writing all the data representing that business process out to disk. At some point, the awaited event occurs. At this time, Ensemble reads the data into an actor job and the business process continues. This means the process isn't affected by a system restart and is using no resources, except for a little disk space.

Development Questions

Q: Why use BPL instead of code?

A: Because it's easy! It's accessible – you don't have to be a programmer to understand what it's doing. You get a visual representation of what's happening in your transformation, instead of having to read through code. A programmer might develop the BPL class and then usefully show it to an analyst with domain-specific knowledge, but no programming skills. You just can't do that if you have written the process in Caché ObjectScript or Java.

Q: Can I use BPMN authoring tools to create BPL?

A: No. In general BPMN tools will generated W3C BPEL and not Ensemble BPL.

Q: How do I put comments in my BPL?

A: You can add 'Annotation' to any activity in your BPL. These are described in the <annotation> page [http://docs.intersystems.com/ens20102/csp/docbook/DocBook.UI.Page.cls?KEY=EBPL_annotation] of the Ensemble Business Process Language Reference.

Q: How do I test my BPL?

A: You can only test BPL as a business process in a running production. If you have enabled testing for the production, you can send a test message to the business process. Select the business process in the configuration page and use the 'Test' button on the Actions tab.

Q: What is the difference between request, response, callrequest, and callresponse?

A: A business process starts with a request and probably sends back a response. Also, the main purpose of BPL is to orchestrate

calls to applications, so you will send requests to other components and get responses. With all these requests and responses, it is easy for a newcomer to get confused but is easy once you get the concepts straight.

The initial request to the business process can be accessed at any time by using the 'request' variable. 'request' is an object reference so if, for example, it has a Name property, that could be referenced as request.Name.

In most cases, the BPL is building a response to send back to the caller. The object reference to this is stored in the 'response' variable and properties of the 'response' object are set as the BPL is executed.

To invoke external functionality, you must send invoke a business operation by using <CALL> activity. Within a <CALL> activity, the request that is being sent is known as the 'callrequest'. This does not exist before the <CALL> activity and won't exist after the <CALL> activity has completed. Part of the call activity is to set the properties of 'callrequest'. Similarly, if a response comes back it is referred to as 'callresponse'. 'callresponse' will also cease to exist when the <CALL> is completed, so any values that are needed must be copied into the context or into 'response' – or the information will be lost.

If you have several <CALL> activities in a BPL diagram, the callrequests and callresponses from different occurrences have nothing to do with each other.

Once again, the variables 'request' and 'response' always refer to the initial request received by the BP and the response that will be sent back to the caller. 'callrequest' and 'callresponse' only exist within a <CALL> and these are the messages sent to and from a Business Operation, or another Business Process.Troubleshooting Questions

Q: If I set a variable in one activity, I can sometimes use it in a later activity and sometimes I can't. Why?

A: A variable will stay in scope from one activity to another but, if the process issues a CALL statement, the whole process is saved to disk and only the response, context, and response objects are saved. When the process is read back into memory, any local variables will be gone. It is best practice not to rely on any local variable being available from one activity to another.

Q: My context properties seem to be truncated at 50 characters – how come?

A: The BPL context is a class like any other and the context properties are properties of that class. The default maximum length of a string property in Ensemble is 50 characters and it will be truncated at that point. To change that, you have to set the MAXLEN parameter for the property. To do this, set the type of the context property to something like %String(MAXLEN=250).

Q: How do I modify the parameters associated with a context property, such as the maximum value allowed?

A: Property parameters can be added to any context property by adding the parameter name and value to the context property type such as '%Integer(MAXVAL=999)'.

Q: Can I 'roll back' changes if one of the applications I'm calling fails?

Q: No, you can't roll back changes but you can use 'compensations' to correct the situation. Every call from a BP is independent (except in the sense of synchronization of calls). When the BP sends a request to a BO (business operation) and that in turn passes the request to an external application, that application will make a complete transaction. Any data inserted into the external application will be available for use. If later in your business process you decide you should not have done so, you have to issue a separate request to deliberately reverse the action. Compensation handlers can be set up in BPL to achieve this, but the details are application specific.

Q: Can I step though my BPL in a debugger?

A: No. Strictly speaking you could connect the Studio debugger to a BP executing some BPL and step through it, but the generated code is complex and you would need a lot of understanding of the internals to make any sense out of it. The only

time you might get any benefit from this would be when you were trying to debug a lengthy <CODE> activity and you should always move significant blocks of code out into external classes that can be independently debugged.

The best way to do this is to add <TRACE> activities and set 'Log Trace Events=true' in the configuration settings for the BP. These trace statements will show up with the existing <CALL> activities in the visual trace.