BEA WebLogic® Integration

BPEL Import and Export User Guide

Version 10.2
Document Revised: March 2008

Contents

CHAPTER 1

Using the BPEL Import Tool

This section describes the procedures to use BPEL Import tool in BEA WebLogic® Workshop to import a BPEL file.

Topics Included in This Section

Introduction to BPEL
Provides a brief background on BPEL and how it evolved.

BPEL Import Tool
Provides an overview on how the BPEL Import tool works.

Importing a BPEL File
Describes how to import a BPEL file using BEA WorkSpace Studio.

Known Limitations and Issues
Provides information on the import tool that will enable you to use is more effectively and efficiently.

Introduction to BPEL

WS-BPEL (Web Services Business Process Execution Language, commonly referred to as “BPEL”) defines a language for the formal specification of automated business processes.

Processes written in BPEL can orchestrate interactions between Web services using XML documents in a standardized manner. These processes can be executed on any platform or product that complies with the BPEL specification. BPEL therefore enables customers to protect their investment in process

automation by allowing them to move these process definitions between a wide variety of authoring tools and execution platforms. While there have been previous attempts to standardize business process definitions, BPEL has attracted an unprecedented level of interest and is the first to gain critical mass among software vendors.

WS-BPEL 2.0 is now an OASIS standard. For more information on BPEL, download the specification from see BPEL4WS specification v1.1, and WS-BPEL specification 2.0. Also, see the official Home page for the BPEL standardization effort, hosted by OASIS. The BPEL import and export tool is provided largely to enable design-time interoperability with other tools that support the 1.1 and 2.0 specifications.

BPEL Import Tool

You can use the BPEL Import tool to import a BPEL file into a JPD file, where it can be used in the BEA WebLogic Workshop design environment. While the main orchestration logic of the BPEL file is imported into a JPD file, it is not expected that the imported JPD file will be immediately executable in Workshop.

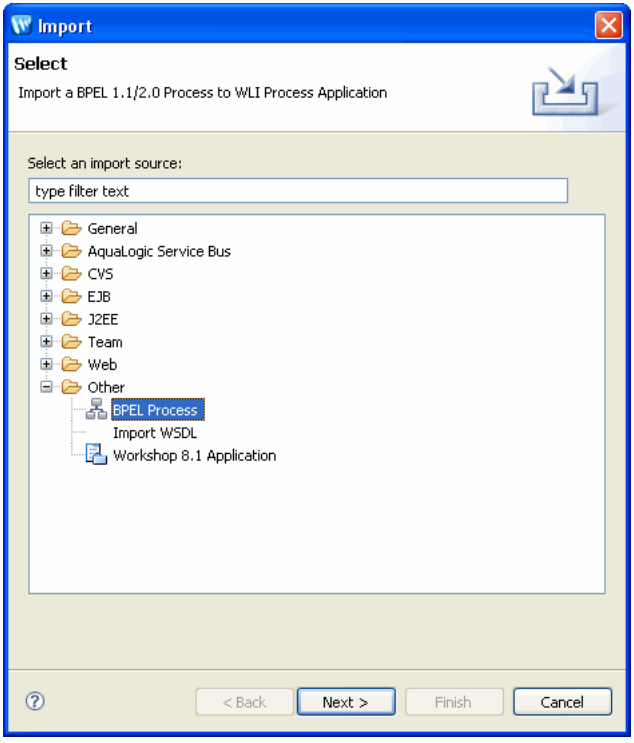
In certain cases, runtime semantics are not guaranteed, due to the functional mismatches between the JPD and BPEL languages, or between various expression languages including differences between XQuery, Xpath, and XSLT. Runtime semantics are also not guaranteed when they involve vendor extensions, external artifacts, or environment settings. Therefore, the imported JPD file should be reviewed and tested with any required changes that are made to ensure that it runs properly.

In general, the BPEL Import tool expects complete BPEL and WSDL artifacts as input. To some extent, the tool also handles incomplete BPEL and WSDL artifacts, so that in-progress BPEL files can be imported as JPD, and then completed in the WebLogic Integration environment. Incomplete cases are numerous and may include missing WSDL files, missing type definitions, missing port type definitions, or incomplete constructs of `<while>`, `<switch>`, `<invoke>`, `<receive>`, `<reply>`, `<onMessage>`, `<onAlarm>`, `<throw>`, as well as other cases. If the BPEL Import is not able to import the input artifacts into a JPD file, error messages appear that enable you to correct the input artifacts for future imports.

Importing a BPEL File

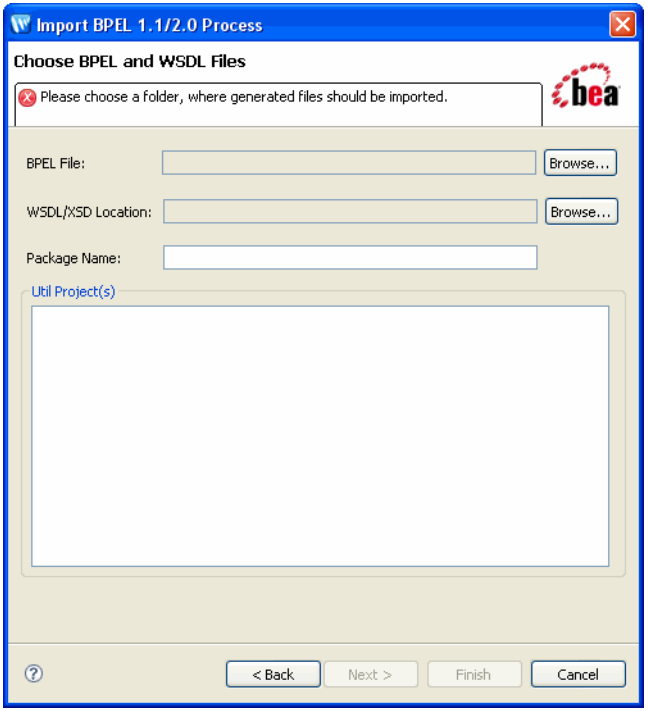
1. In BEA Workshop for WebLogic, create or open a BEA Workshop for WebLogic application.
2. In the **Package Explorer** pane, navigate to **Web Project** > **Src > Import** to import the BPEL file, as shown in Figure 1-1. The **Import** pane is displayed.

Figure 1-1 Location for Imported File



Note: The BPEL Import tool does not support importing to Schemas/Project root directories. If you try to import into a root directory, you get the following error message Figure 1-2.

Figure 1-2 Error Message

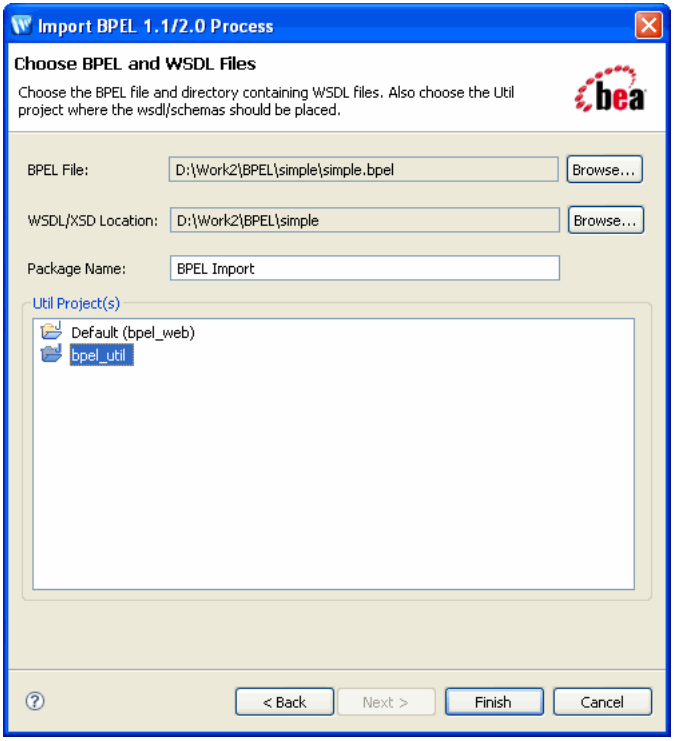


3. To choose the files to import, browse to the **BPEL File** folder, to select the required `.bpel` file. Similarly browse to the **WSDL and XSD Locations** to set the path for Web Services Description Language WSDL and XSD files as shown in Figure 1-3.

   Type an appropriate Package Name in the **Package Name** Field..

   Select the **Util project** folder. Preferably the make it point to the Util project created while reacting the Weblogic Process Application.

Figure 1-3 Import BPEL Source Window



4. You can view the log containing the complete details of the import as shown in Figure 1-4. This log also tells you if the import was successful or not.

Figure 1-4 Import log

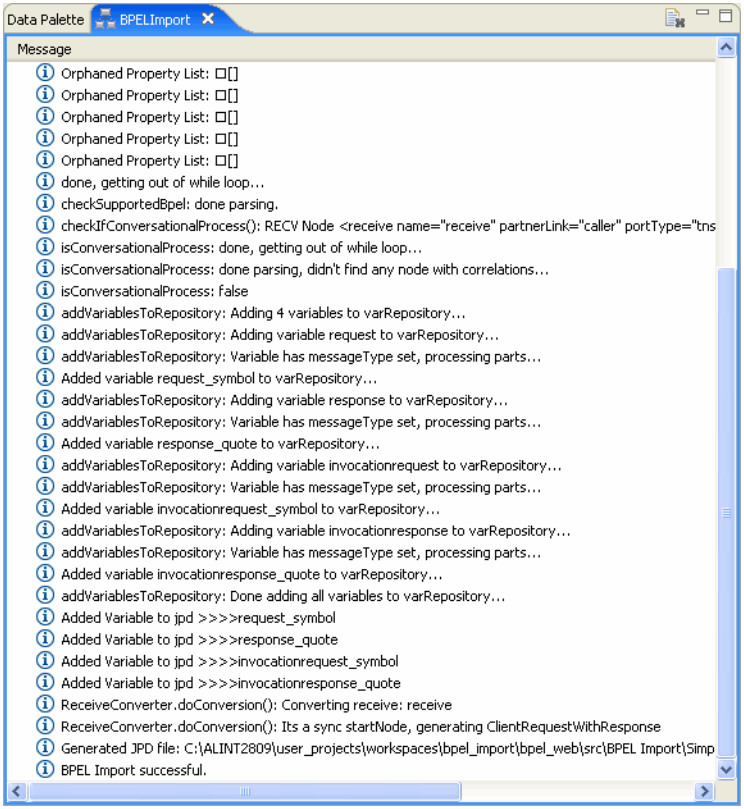
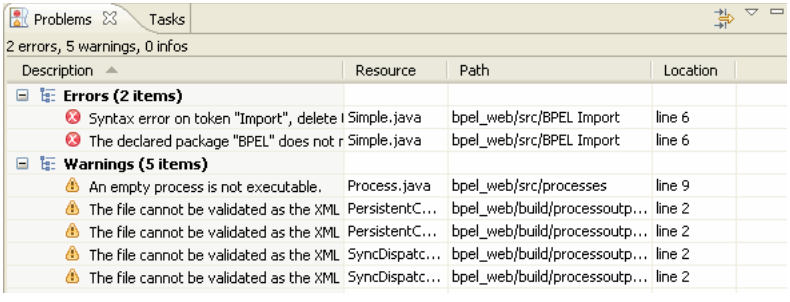


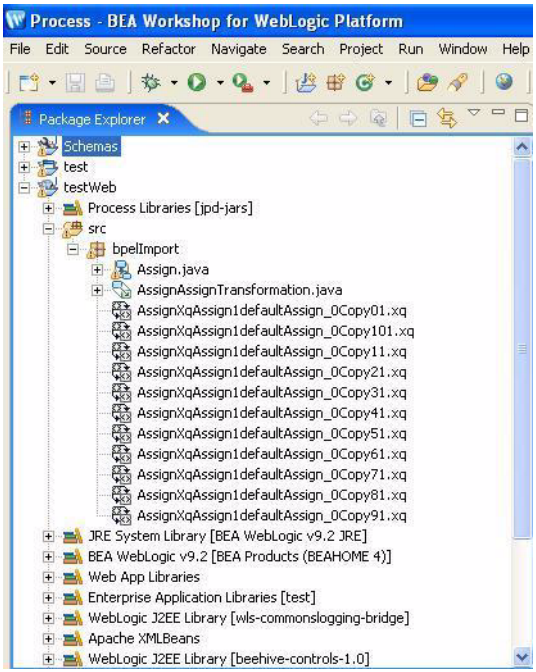
Figure 1-5 Problem View - Warning Messages



Note: A log file for the import process, named `BpelImport.log` is stored in the **workspace/.metadata** folder (workspace indicates your Weblogic Process Application workspace).

The new JPD file is displayed in the folder specified in step 2 on page 1-2 and the WSDL files are copied and placed in the Schemas folder, as shown in Figure 1-6.

Figure 1-6 New JPD and WSDL Files



This completes the import process and your new JPD file is located in the folder specified in step 2 on page 1-2.

Known Limitations and Issues

This section details some known limitations and issues of the BPEL Import tool. The majority of these issues exist because of the inherent differences between the JPD and BPEL languages.

Note: You need to confirm that the generated JPD file corresponds semantically with the input BPEL file.

- Conversion of Compensation Handlers is not supported. If you try to convert a BPEL file that contains Compensation Handlers, a warning message is displayed stating that

Compensation Handlers are not supported and are ignored in the generated JPD file. The conversion process will continue after this warning is displayed.

- Global `eventHandler` is not supported. If it is included in the BPEL file, it is ignored during the conversion process and the following message appears: `Global EventHandlers are not supported, hence ignored`.
- Conversion of `wait` and `onAlarm` for which the duration is specified using the `until` attribute is not supported. If the `until` attribute is contained in the BPEL file, a warning appears stating that it is not supported and is ignored in the generated JPD file. The conversion process will continue after this warning is displayed.

Note: Both attributes (`for` and `until`) cannot be specified in a valid BPEL file.

- A BPEL file that starts with a `flow` is not supported. You cannot convert a BPEL file that has a `flow` construct as a first logical child. In this instance, logical activity refers to any activity other than sequence and scope.
- Conversion of `links` from activity `flow` is not supported. If `links` are present in the BPEL file that you want to convert, a warning appears stating that the generated JPD file may be erroneous as it contains `links`, `source`, `target`. The conversion process will continue and will ignore these unsupported activities.
- Conversion of a BPEL file in which more than one `reply` activity is used to return normal output/outcome is not supported. In order to qualify for conversion, a BPEL file must only contain one `reply` activity to return a normal outcome. This is due to the fact that in a JPD file, there can only be one `returnMethod` for any synchronous `clientReceive`. However, in a BPEL file, there can be several `reply` activities for a single `receive`. There is no direct way to map one `receive` and several `reply` nodes to a single `clientRequestWithReturn`.
- A `pick` activity with more than one `OnAlarm` activity is not supported for conversion. This is due to the fact that `pick` gets converted to `eventChoice`. Each `eventChoice` can have at most one `timeoutEvent` node, which is generated for an `onAlarm` activity.
- Specifying the `for` attribute of `onAlarm` activity using `bpws:getVariableData(..)` is not supported. When a `for` attribute is specified using `bpws:getVariableData(..)`, the imported code produces a syntax error.
- If the import fails with the following message: `WARNING: Failed to parse input XSD & WSDL files. Please see logs for detail`, and the log file contains the following error message: `Duplicate global type`, you should specify that multiple definitions of the namespace are ignored.

The log file for the import process, named `BpelImport.log`, is stored in `BEA_HOME\user_projects\workspaces\bpel_import\.metadatap` where `%BEA_HOME%` is the product home directory. This log file provides information about the import process.

To specify that multiple definitions of the namespace are ignored, select **Schema Project→ Properties→ Build** and list the namespaces to be ignored in the **ignore multiple definitions in following namespaces** field.

- Limited query parsing is performed when `assign` activities are imported. Therefore, the generated XQuery expressions might not always be syntactically correct. Ensure that you check to verify the correctness of any generated XQuery expressions.
- Any reference to soap encoding types in your BPEL file will result in import failure. If you want to import a BPEL file that contains soap encoding types, you must place the *SOAP Encoding* XSD file in the WSDL directory when importing through BEA Workshop for WebLogic.
- When `assign` is converted to a variable, the resultant XQuery file may contain the following "PARSE ERROR" in the Design view: `The main XML element does not match the root node of the target schema`. However, you can ignore this message as the correct value is generated in runtime. This error is due to the fact that the **Design** view of XML Mapper has limitations and might not always be able to parse even correct XQuery expressions.
- After you import a BPEL file that contains scope level `eventHandlers` in the onMessage branch, a dummy Timer method may be generated for the `eventHandler` onMessage path. JPD specification mandates that `onMessage` and `onTimeout` paths can only be associated with process nodes (or block of nodes) that do not run automatically. To handle this constraint, a dummy Timer node with a 1 second timeout is created if an `eventHandler` is associated with a scope that does not contain any `receive`, `onMessage`, `flow`, or `wait` activity.
- When a BPEL file is imported, queries are not parsed, they are imported 'as is'. You must properly qualify the query string or the generated XQueries will fail at runtime, by setting the `attributeFormDefault="qualified"` and `elementFormDefault="qualified"` and then using the qualified query string.
- A BPEL file may have `reply` activities on multiple partner links. This is not supported in a JPD file. Only one such partner link will be converted as `clientRequestWithReturn` that matches the reply semantics. `reply` activities on other partner links will be converted into an asynchronous interface.
- BPEL Event Handlers are translated to JPD `onMessage` paths which have slightly different semantics. In the imported JPD file, once the event is received and the `onMessage` path is executing, the block (corresponding to the BPEL scope) does not continue executing until the

onMessage path completes. This also means that only one instance of an Event Handler can be active at any one time.

- In the imported JPD file, Event Handlers will not be triggered while waiting for the response of a synchronous Web service.
- The BPEL Import tool converts BPEL invoke activities to Web Service controlSend calls. It generates a Web Service control for the specified partner service and the JPD file invokes the corresponding method on the control. This release has some limitations in its capability to generate a Web Service control (a JCX file) in certain situations. You should carefully examine the contents of the generated JCX file to ensure that you can compile it.
- If the input WSDL file does not have any bindings and service information for any portTypes, the BPEL Import tool tries to generate a method for each portType with some default values for the binding and service information. You should review these default values carefully. You should provide valid binding and service information after the import.
- The JPD file generated from BPEL processes that contain <reply> in Switch, Case, o Otherwise nodes may not be logically correct. This is because the matching receive and reply of the synchronous operation are not present at the same level.
- BPEL allows event handlers to be associated with any arbitrary activities. However, a JPD file does not support cases where event handlers are associated with any node inside a <receive> - <reply> block. In such a scenario, the generated JPD may not compile.
- The BPEL Import tool generates Workshop Web Service Control for external services specified as partnerlinks. However, BEA Workshop Web Service Control does not successfully resolve external schemas imported using relative URIs into the WSDL (associated with the Service Control). You should copy the external schemas from the utility project to the web project and place the external schemas relative to the location of the WSDL in the web project.
- Termination handlers not supported and will be ignored. ExtensionActivity is not supported
- For EventHandlers, attributes until and repeatEvery are not supported and will be ignored.
- BPEL process attributes namely suppressJoinFailure and exitOnStandardFault will be ignored.

Using the BPEL Export Tool

This section describes how to use the BPEL Export tool in BEA Workspace Studio, to export BPEL 1.1 and 2.0 compliant code from a JPD file.

Topics Included in This Section

BPEL Export Tool
Provides an overview on how the BPEL Import tool works.

Exporting a BPEL File to JPD
Describes how to export a BPEL file using BEA Workspace Studio.

Known Limitations and Issues

Provides information on the export tool that will enable you to use is more effectively and efficiently.

BPEL Export Tool

You can use the BPEL Export tool to export the semantics of a JPD file into BPEL where it can be used in a BPEL design environment. BPEL code that is exported using the BPEL Export tool is BPEL 1.1 and 2.0 compliant and can be used in design environments compliant with BPEL 1.1 and 2.0. While the main orchestration logic of the JPD is exported to BPEL, it is not expected that the exported BPEL will be immediately executable in the target environment.

This is due to the fact that some executable call-outs from the JPDs will be opaque to the exported BPEL code. These executable units generally include controls, code written in perform nodes, and XQuery transformations. The BPEL Export tool copies the Java code and the XQuery code as extension

nodes in BPEL. As a result, you must re-implement the logic in the target BPEL environment, as JPD provides a superset of the functionality provided by BPEL.

One Web Service Definition Language (WSDL) file defines the WSDL interface of the business process and defines a partner-link type for the interface. The other file defines the WSDL interface and partner-link types of the partners. Partners are the artifacts interacting with the business process. These artifacts are either consumers or providers of services to the business process.

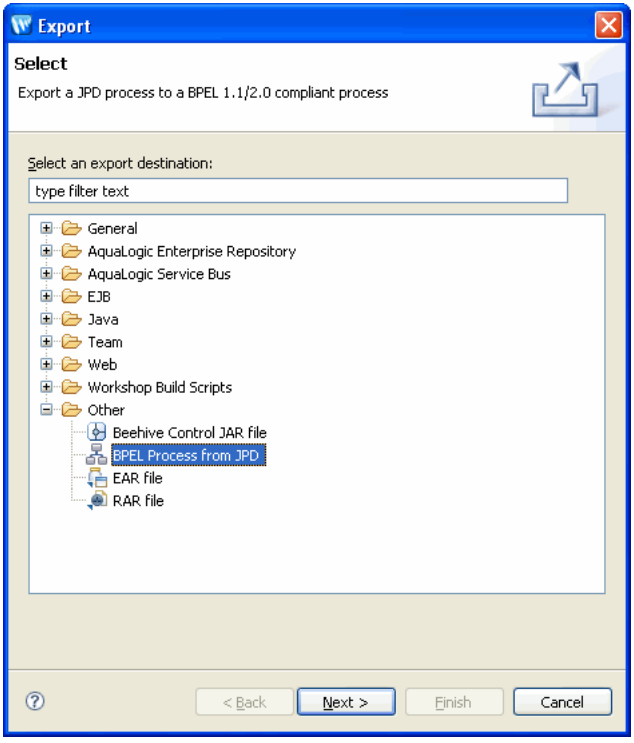
These WSDL files are not the same as the WSDL that BEA Workspace studio would generate for the corresponding JPD or JCX files. The differences are described in detail in "Known Limitations and Issues" on page 2-7.

The relevant XSD schema files (which must be located in a schema folder in the WebLogic Integration application) are needed in the target environment, along with the WSDL and BPEL files.

Exporting a BPEL File to JPD

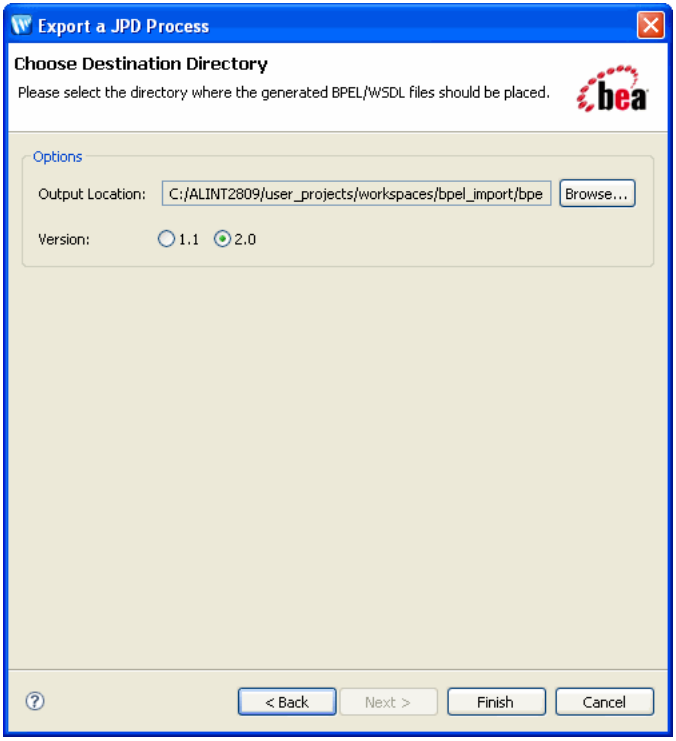
1. In BEA WorkSpace Studio, right click on a JPD file in the **Package Explorer** pane. For example, right click on `Process.java` **Export**.
2. The **Export** dialog box is displayed. From the **Select an export destination** pane, expand **Other**, and select **BPEL Process from JPD** as shown in Figure 2-1 and click **Next** to proceed.

Figure 2-1 Export Option



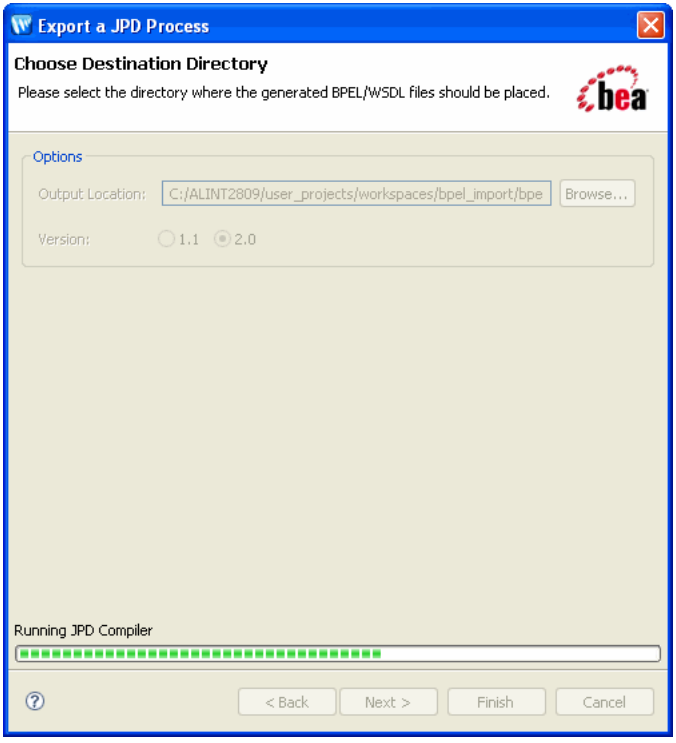
3. Choose the destination directory where the BPEL Export tool will create BPEL and WSDL files as shown in Figure 2-2.

Figure 2-2 BPEL Export Pane



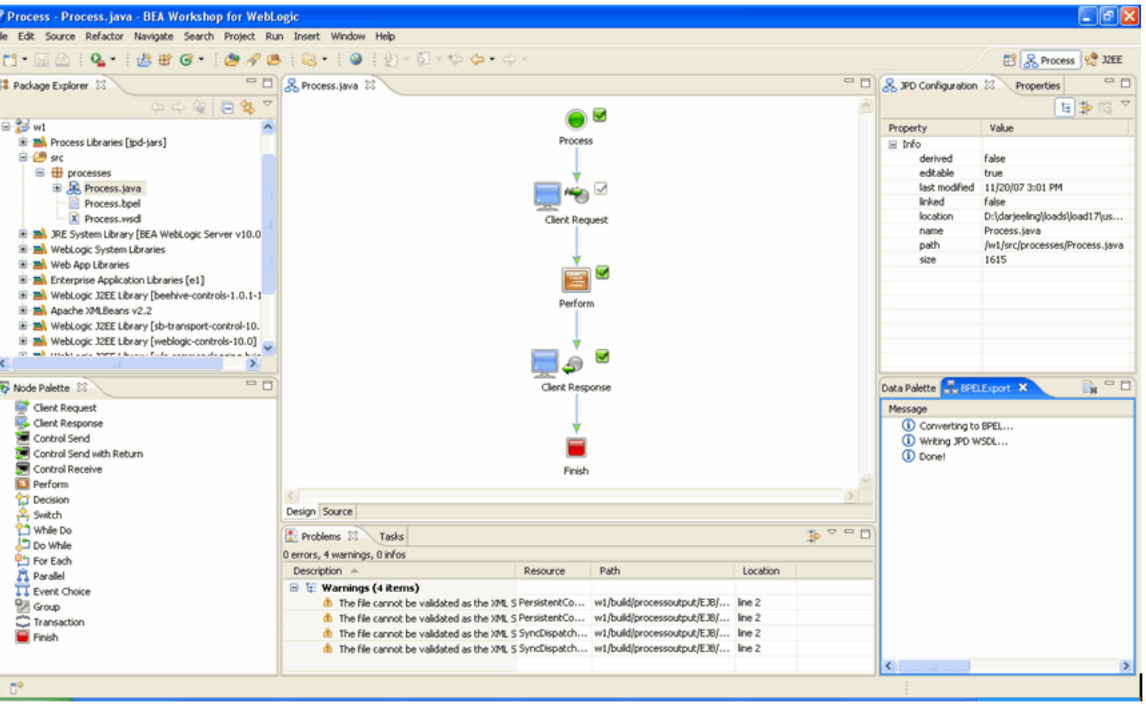
4. Select **Finish** to complete the export process. You can view the progress of the export as shown in Figure 2-3.

Figure 2-3 Progress of BPEL Export



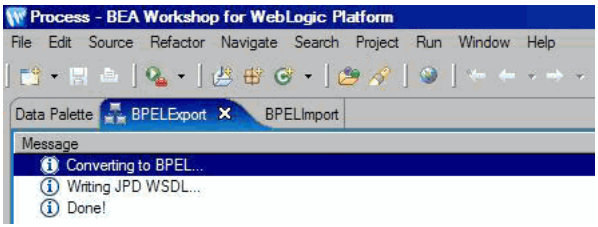
5. When the export is complete, the BPEL Export log displays the message **Done!** as shown in the lower right hand corner of Figure 2-4.

Figure 2-4 Generated Files



6. You can view the log containing the complete details of the export as shown in Figure 2-5. This log also tells you if the export was successful or not. The logs are located in `%workspace%/.metadata` where `workspace` refers to where the WebLogic Process application resides.

Figure 2-5 Export log



Known Limitations and Issues

This section provides some notes on the export process and details some known limitations of the BPEL Export tool.

Notes: The section details information that you should remember when using the BPEL Export tool.

- Process variables in a JPD file are converted to variables in the outermost scope of the BPEL file. `int`, `short`, `long`, `byte`, `float`, `boolean`, `double`, `String`, `java.util.Date`, and `java.util.Calendar` are converted to the corresponding schema built-in types. XMLBean types are converted to the corresponding XML Schema type. A wrapper element type is introduced for complex types, BPEL variables cannot have a complex type as their type. Any other types (including temporary transform variables) are converted to an element type with no type attribute.
- Controls are exported as partner-links. The operations for this partner-link are derived from the methods in the control JCX file. Each method parameter is treated as a separate input message part; the name of the part is the same as the name of the parameter. The output message is determined from the return type of the control method. It has a single part called `parameters`, since a method has a single return type with no name.
- XQuery expressions in the JPD file are copied 'as is' into the expression attributes of the corresponding BPEL activity. XQuery code referenced in a Data Transformation control is copied into the JPD namespace `xqueryCode` element.
- Before and after transformation, `send` or `receive` messages are exported to assign activities before or after the corresponding `invoke`, `reply`, or `receive` activities.

Table 2-1 details how various JPD file attributes, nodes, and so on are converted to a BPEL file.

Table 2-1 JPD File to BPEL File Conversion

In the JPD file	converted to in a BPEL file
`clientRequest` and `controlReceive` nodes	`receive` activity
`clientCallback` and `controlSend` nodes	`invoke` activity
synchronous `returnMethod` attribute	`reply` activity
`onMessage` paths	`onMessage` event handlers
`onTimeout` paths	`onAlarm` event handlers

Table 2-1 JPD File to BPEL File Conversion

In the JPD file	converted to in a BPEL file
`eventChoice` node	`pick` activity
`Parallel` node	`flow` activity
`onException` blocks	`faultHandlers`
`perform` node	`empty` activity with the java code from the `perform` method copied into the body of a JPD namespace javaCode element
control flow nodes: • `decision` • `switch` • `forEach` • `doWhile` • `whileDo`	to the equivalent BPEL activities.

- Any warning messages that are generated during the export process do not include an exact line number reference of the original JPD file.
- Constants declared in a JPD file are captured in the BPEL file as `jpd:initialValue`.
- When you export a JPD file that contains a Service Broker control, the shape of the process and control WSDLs are derived from the JPD file and are different from the JPD WSDL.
- Any methods which are not directly associated with a JPD node are lost during the export.
- If a converted control (service or process) produces a method with a void return, there are two possible causes:
  - the corresponding operation has no output message.
  - the operation has an output message with no parts.

  The BPEL Export tool does not distinguish between the two cases and always assumes that the first case is true.
- User schemas are referenced by a `xsd:include` element. If the types used are in a WSDL file, it is exported using `wsdl:import`.

- If the schemas folder is created with the default name, all XSD files placed in the top level directory of this folder will be referenced by an absolute file URL. For all other XSD files, just the filename is referenced.
- If a JPD file contains `ArrayList` or another Collection class's `add()` method, a non-standard JPD namespace attribute `jpd:appendToCollection` is generated with its value set to `true`.
- If MFL types are used in the JPD file, they are converted to a dummy empty type. A warning to this effect is emitted. The warning message is: `MFL types are not supported for export. Creating an empty element type for <type>`.
- Assign statements are generated to assign global variables to and from Web service messages. However, if the Web service message does not have any part defined, no assign statement is generated.
- Service controls are treated as generic Java controls. The original WSDL file for the Service control is not used in the export.
- `afterExecute=resume` is not supported for the following paths:
  - `OnTimeout`
  - `OnException`
  - `OnMessage`
- `freezeOnFailure=true`, `onSyncFailure`, and `persistent` are not supported.
- `executeOnRollback` is not supported for `OnException` path.
- Transaction blocks are converted into a scope with BPEL extension `jpd:transaction` set to `true`.
- XQuery transformations are copied into the `<jpd:xquerycode>` node as a BPEL extension.
- During the export process, Java code is copied into the `<jpd:javacode>` node as a BPEL extension.
- Message Broker subscriptions are exported as partner links. The message broker channel name and subscription filter are not included in the export.
- Perform nodes are exported as an empty activity. Java code is included as a `<jpd:javacode>` extension.
- Attribute information for controls in the JCX file or before the control declaration are lost. For example, for Message Broker controls, the channel name and subscription filters are not copied into the exported BPEL or WSDL files.

- When process variable names, control file names, control method names, parameters used in control methods, variable names defined for controls, and `.java` file names that contain special character like `$` are exported to `.bpel` files, these names are used as is in the “name” attribute of “variable” and “variable” attribute of “to”. Since BPEL schema defines these attributes as NCName type, these special characters become invalid in the generated `.bpel` file. However, this limitation is no longer valid for the `$` character. For any other special character (that is not valid NCName or QName type), althought the .bpel file is generated, schema validation of the file fails.

  Workaround: The generated `.bpel` file needs to be manually modified by using valid characters.

APPENDIX A

BPEL Process Compatibility Limitations between WLI and ALBPM

Importing WLI Generated BPEL 2.0/1.1 processes into AquaLogic Business Process Management

The BPEL process generated by WLI 10.2 cannot be directly viewed in ALBPM 6.0. The following are the known workarounds to be applied for importing a BPEL process generated by WLI into ALBPM.

1. BPEL process in ALBPM uses `http://schemas.xmlsoap.org/ws/2004/03/business-process/` namespace. WLI generated BPEL 2.0 process uses `http://docs.oasis-open.org/wsbpel/2.0/process/executable` namespace. WLI BPEL 1.1 uses `http://schemas.xmlsoap.org/ws/2003/03/business-process/`. Replace `http://docs.oasis-open.org/wsbpel/2.0/process/executable` *with* `xmlns=http://schemas.xmlsoap.org/ws/2004/03/business-process/'>` and `http://schemas.xmlsoap.org/ws/2003/03/business-process/` *with* `xmlns=http://schemas.xmlsoap.org/ws/2004/03/business-process/'>`.

   Note: The namespace information for BPEL 2.0 is available at: http://docs.oasis-open.org/wsbpel/2.0/process/executable. BPEL 1.1 information is available at :`http://schemas.xmlsoap.org/ws/2003/03/business-process/`.

2. In the BPEL process, modify the `expressionLanguage=urn:bea:wli:bpm:wsbpel:2.0:sublang:xquery1.0` to `expressionLanguage=http://www.w3.org/TR/1999/REC-xpath-19991116`.

3. Include `<Import>` element.

4. WLI generated BPEL extensions such as `<jpd:javacode>//</jpd:javacode>`, `<jpd:xquerycode>`, `jpd:transaction`, `jpd:name`.
5. In a WLI generated BPEL, delete any special characters from the names of constructs.

Importing ALBPM Generated BPEL 2.0/1.1 processes into WLI

For using the BPEL processes from ALBPM in WLI, replace the namespace with `http://docs.oasis-open.org/wsbpel/2.0/process/executable` for BPEL 2.0.

For BPEL 1.1 replace the namespace with: `http://schemas.xmlsoap.org/ws/2003/03/business-process/`.