WebSphere Process Server Versioning: From Design to Production

Introduction

Taking a WebSphere® Process Server application involving long-lived processes from design to production requires consideration and planning for many aspects to be successful. Designing a process versioning scheme is one of those aspects. Without a well-planned process versioning scheme, you might be prevented from introducing new functionality into production, and the Software Development Lifecycle (SDLC) costs will be higher.

This paper discusses certain design and packaging considerations for implementing an efficient process versioning scheme that provides maximum flexibility in terms of changing functionality of an application already in production.
Versioning and long-running processes

In WebSphere Process Server, long-running processes (or macro flows) are the main driver for designing a versioning scheme. The reason, as you already know, is because active macro flow processes are persisted in the BPEDB, and the application EAR that was used to instantiate them cannot be removed until all active (or in-flight) processes associated with the EAR have completed.

This means that if you want to change a long-running process, you have to use the process versioning capabilities of WebSphere Process Server.

Late binding and process templates

The method that WebSphere Process Server uses for versioning is to allow the creation of multiple versions of a process template. Each process can have many process templates, and each one will have the same process template name but different valid from dates. When a process instance is instantiated (created) for the first time, all of the available process templates will be found (in the BPEDB), and the one with the valid from date closest, but not past, the current date and time will be used. The process templates deployed can be seen using BPC Explorer (Figure 1).

![BPC Explorer v7 showing process templates](image)

Figure 1  BPC Explorer v7 showing process templates

**Note:** You can only view the process templates with the latest valid from dates with the BPC Explorer V6 and later.
Existing process instances are always tied to the process template that they were first instantiated with, and this is immutable. Also, the application EAR that contains the process template used to instantiate the long-running processes cannot be removed from WebSphere Process Server (neither an uninstall nor an update of the application is possible) until all of the long-running processes run to completion or are terminated.

To enable process versioning, WebSphere Process Server allows late binding of process templates so that the latest version of the process template can be used when a process is instantiated. Late binding means that the process template to be used for instantiating a process is determined at run time by the Business Process Choreographer (BPC) engine. To use late binding of processes, the process template name to use needs to be specified in the target process attribute (Figure 2) of the corresponding reference partner in the BPEL process (there is a reference partner in a BPEL process for every external import invoked from the BPEL process). So when the BPEL process calls out to another BPEL process through a reference partner with late binding configured, a runtime search will be made of all process templates with the name specified in target process attribute of the reference partner, and the one with the latest valid from date will be used to instantiate the process.

![Figure 2 - BPEL Properties dialog in WebSphere Integration Developer showing target process edit box](image)
Effects of packaging on versioning

Process versioning considerations influence how applications are packaged into individual modules.

The goal is to package your application so that the inevitable changes that you will make to your application affect as few modules (and therefore EARs) as possible.

Packaging SCA Java components

If an SCA Java™ component is called by more than one long-lived BPEL process, the SCA Java component should not be packaged with any of the long-lived BPEL processes. The reason is that if the SCA Java component must change, only its module must change.

Additionally, segregate your shared SCA Java components that are functionally related into one module and expose them with SCA Exports (or other binding types). SCA Java components are similar to micro flows in that they are not persisted in the BPEDB.

By packaging functionally related SCA Java components in the same module, it is easy then to change their implementation and re-deploy them. None of the other calling modules will require versioning (assuming that the interfaces remain the same). The only disruption is that the server might require a restart.

Packaging micro flows

BPEL micro flows that help implement a single functionality should be grouped together in one module and exposed with SCA (or other binding type) exports. BPEL micro flows are not persisted in the BPEDB. By packaging them in the same module, it is easy then to change their implementation and re-deploy them. None of the other modules are required to be versioned (assuming that the interfaces remain the same). The only disruption is that the server might require a restart.

All micro flow components that are called by more than one long-lived BPEL process should be packaged separately from the calling BPEL processes.

Packaging macro flows

BPEL macro flows always need to be versioned because their process instances are persisted in the BPEDB (long-lived), as explained earlier. Packaging each
macro flow in separate modules gives the most flexibility because you are not forced to version unmodified BPEL processes that happen to be packaged in the same modules as BPEL processes being modified. However, consider how many EARs you want to manage (the more EARs the more EARs you must build and deploy). The decision to package macro flows together in a single module should be made taking into consideration the fact that any change to the application implementation results in process versioning of all the macro flows contained in the EAR.

Consider the following scenario:

- BPEL component A is a long-lived process making a call to POJO X and Mediation M.
- BPEL component B is another long-lived process that only makes a call to Mediation M.

Table 1 lists the packaging options and their advantages and disadvantages for the above scenario.

<table>
<thead>
<tr>
<th>Packaging option</th>
<th>Advantages and disadvantages based on degree of flexibility for process versioning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package all components in a single module.</td>
<td>Any change to BPEL component A (that is, versioning of BPEL component A) necessitates versioning of BPEL component B. Therefore, this is a poor packaging choice if versioning of BPEL component A and BPEL component B does not necessarily happen simultaneously. Also, any change to the Mediation M or POJO X forces a process versioning of both BPEL component A and BPEL component B, so this is the worst packaging choice from a process versioning viewpoint.</td>
</tr>
<tr>
<td>Package A, X, and M in one module and B in a separate module.</td>
<td>If Mediation M or POJO X needs modification, then this packaging option forces process versioning of BPEL component A. In addition, though not very obvious, this option also has the same drawbacks of the first option above because BPEL component B makes a call to Mediation M. So if BPEL component A is versioned or if Mediation M is changed, the binding of the SCA import of Mediation M that has to be in BPEL component B’s module needs to be changed to point to the new module (which needs to be created due to versioning of BPEL component A). So this packaging is also not optimal.</td>
</tr>
</tbody>
</table>
As a leading practice, only those components that are called *only* by a macro flow contained in a module should be considered for inclusion into that module. Any component that is being called by other BPEL components from different modules need to be packaged in a separate module.

### Keeping Java snippets small

There is nothing wrong with Java snippets per se. However, keep the amount of Java code small in a Java snippet. Instead, package the Java code in a stand-alone JAR project and just call the Java code (in the JAR) from the Java snippet. This technique allows you to deploy the JAR in an external directory (via a WebSphere Classpath variable) or in a directory on the WebSphere classpath (such as \lib\ext). Then you can make changes to your Java code, package your new JAR file, stop the server, deploy the JAR file, and restart the server. Again, none of the other EARs will require a change.

### Packaging Java utility projects

If a Java utility project is embedded (deployed with a module) in an application containing a long-lived process, then any change to the Java utility project forces versioning of all the long-lived processes in a given module. To avoid this, consider the following packaging options:

<table>
<thead>
<tr>
<th>Packaging option</th>
<th>Advantages and disadvantages based on degree of flexibility for process versioning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package A and B in one module and X and M in another module.</td>
<td>This packaging might not cause maintenance problems, if BPEL component A and BPEL component B are tightly coupled in terms of versioning (that is, a change to BPEL component A necessitates a change to BPEL component B and vice versa). Otherwise, this option is not optimal, as it forces versioning of both BPELs whenever there is a change to one of them.</td>
</tr>
<tr>
<td>Package A and B in separate modules and X and M in another module.</td>
<td>This is the optimal packaging for versioning, as any change to one component does not force versioning or change to another.</td>
</tr>
<tr>
<td>Package all components in separate modules.</td>
<td>This option also can be used when packaging for versioning. The number of application EARs increases though.</td>
</tr>
</tbody>
</table>

Note: If any shared component is included in the module containing a macro flow, then all the components that call this shared component have to be versioned when the macro flow is versioned.
unnecessary process versioning, any Java utility project intended for use by a long-lived process or by components packaged in the same module as a long-lived process should not be deployed with the module. Instead, use the Java utility project as a shared library.

**Note:** Certain constraints are applicable when using a Java utility project as a shared library. For details see the technote at the following website:


**Starting workflows through a version façade micro flow**

You should always have a BPEL micro flow that acts as a façade between your client application and the BPEL modules containing your workflow logic if the client application does not have process-versioning capabilities. This allows you to decouple BPEL versioning changes from your client application. For example, consider the scenario in which a client application (a UI application using a stand-alone reference or a POJO) needs to start a workflow in WebSphere Process Server by instantiating a long-lived process. If the client application has code to directly instantiate the long-lived process, then the client application always will have to be changed when the long-lived process is versioned. Instead, having the client application call a façade micro flow, which in turn calls the long-lived process via late binding, avoids changes to the client application when the long-lived process is versioned.

**Including library projects in your modules**

Because library projects can change and many of the artifacts stored there are interface artifacts (WSDL and XSD), package them with the BPEL modules that use them. This allows you to make changes to library projects without worrying about process templates with earlier *valid from* dates, because they will be tightly coupled with the versions of the library modules that they were built to use.

To be clear, it is not a leading practice to package implementation code (such as Java code) in SCA library projects either. The code should be packaged in its on project.
Limitations of process versioning

When making changes to an application, the following must be observed for process versioning to be effective:

- No namespace or signature changes for the interfaces should be implemented by long-lived processes.
- No namespace change should be made to the business objects used by the interfaces that are implemented by long-lived processes.
- No change should be made to the correlation set or correlation properties used by the long-lived processes.
- Other changes to the business objects should be backward compatible, and only optional attributes should be added to the business objects used by the long-lived processes.

Any change that does not conform to the limitations above should be a candidate for a new application implementation.

WebSphere Integration Developer V7 versioning wizard

WebSphere Integration Developer V7 has a new versioning wizard. In earlier versions of WebSphere Integration Developer, you had to manually copy and paste a new version of your WebSphere Integration Developer project with a new name. Then you had to open the new project and manually update every valid from date in every BPEL process. After that, you could make the required functional modifications.

The new WebSphere Integration Developer versioning wizard automatically creates a new version of a project and updates all of the valid from dates.
When you right-click a project, you will see the context menu. Select **New Process Version** (Figure 4 on page 10).

*Figure 3  Context menu for Business Integration projects*
On the next panel you can specify the name of your new project, or you can accept the default name (the wizard properly increments the version number) and click **Next** (Figure 5 on page 11).

---

**Figure 4**  Versioning wizard (first page)
On the next panel you can specify the valid from date to use. Every BPEL process (that you chose) in the project will be updated with this valid from date (Figure 6 on page 12).
In the next panel you can specify which BPEL processes should have their valid from dates updated (Figure 6). Make your selections and click Finish.

Figure 6 Versioning wizard (third page)
After the wizard finishes, you can make sure that the BPEL processes have the correct valid from dates by checking the BPEL process Properties dialog (Figure 7).

![Properties dialog](image)

**Summary**

This paper presented various aspects to consider for process versioning a WebSphere Process Server application. It covered ideas on how best to package the application into modules, how to use late binding, and how to edit application artifacts for versioning.

**Further information**

- WebSphere Process Server V6.2 Information Center Versioning Processes
  

- WebSphere Process Server V7 Information Center Versioning Processes

The team who wrote this paper

This paper was produced by a team of specialists from around the world:

**Owen Cline** is a member of the IBM® Software Services for WebSphere team based in San Diego, CA. He earned a bachelor’s degree in computer science from the University of Pittsburgh and a master’s degree from San Diego State University. He has over 20 years of experience in the software development field. He holds four software patents, has written IBM Redbooks® publications, and has presented at multiple technical conferences. For the past five years, Owen has specialized in SOA and J2EE architecture and application development and deployment with a special emphasis on the WebSphere platform. In addition, he has worked on many high-profile websites over the past few years.

**Mahesh Surya** is a Managing Consultant in the IBM Software Services for WebSphere team and is based in Rochester Hills, MI. He has over 13 years of experience in IT and has specialized in process integration with an emphasis on WebSphere Process Server. He has also worked in SAP customization and integration projects.

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

[ibm.com/redbooks/residencies.html](http://ibm.com/redbooks/residencies.html)
Stay connected to IBM Redbooks

- Find us on Facebook:
  http://www.facebook.com/IBMRedbooks
- Follow us on Twitter:
  http://twitter.com/ibmredbooks
- Look for us on LinkedIn:
  http://www.linkedin.com/groups?home=&gid=2130806
- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:
- Stay current on recent Redbooks publications with RSS Feeds:
  http://www.redbooks.ibm.com/rss.html
Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:
This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

© Copyright International Business Machines Corporation 2011. All rights reserved.
Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
This document REDP-4724-00 was created or updated on January 21, 2011.

Send us your comments in one of the following ways:
- Use the online Contact us review Redbooks form found at: ibm.com/redbooks
- Send your comments in an email to: redbook@us.ibm.com
- Mail your comments to:
  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099, 2455 South Road
  Poughkeepsie, NY 12601-5400 U.S.A.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the web at http://www.ibm.com/legal/copytrade.shtml

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

IBM®
Redbooks®
Redpaper™
Redbooks (logo) ®
WebSphere®

The following terms are trademarks of other companies:

Java, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.