

A Closer look at WBI-ICS and WebSphere Process Server

[For customers migrating from WebSphere Interchange Server (ICS) to WebSphere Process Server]

WebSphere Process Server, the next generation of business integration software from IBM, is based on a service-oriented architecture (SOA) that embraces open standards and is also the recommended migration target for existing WebSphere InterChange Server customers. WebSphere Integration Developer v6.0, based on Eclipse 3.0, is the Integrated Development Environment (IDE) that complements WebSphere Process Server and introduces a whole new paradigm of component-based development.

Problems/Opportunity

If your organization runs the old CrossWorlds product or WebSphere WBI ICS product you have some serious questions. This paper was written to assist in your evaluation efforts of an inevitable migration path. It's okay, thousands of customers are in the same boat. For many of those customers the benefits of the WebSphere Process Server outweigh the cost and frustration of the migration. We'll talk about benefits a little later.

WPS is the next rendering of the WBI-Server Foundation. The WebSphere Integration Developer tool set "WID" within WPS is completely different from ICS tooling. ("WID" is the most advanced intuitive efficient development tool we've seen) The WID provides a single development platform without having to navigate multiple development tools, it's a single tool to develop integration artifacts!

Evaluate the migration path; pains, challenges and benefits.

Common Questions: What are my options? Should I migrate to WPS? How does it effect my Message Broker environment, do I need to migrate? Will I save on administration cost/resources? Can we now address some of our architecture/design flaws and create more of a web-services SOA based approach? Are there tools to help me with this migration? What will business owners think? Can this migration help in the definition of our "CoE" Center or Excellence or ICC Integration Competency Center? What will happen with my legacy WBI adapters?

Moving to WebSphere Process Server...

PROS – IBM Future Direction	CONS – Not the Future Direction
<ul style="list-style-type: none"> • You can migrate WebSphere Interchange Server solutions to WebSphere Process Server by converting the WebSphere Interchange Server source artifacts into WebSphere Process Server source artifacts that perform equivalent functions. The migration function provided in WebSphere Process Server and WebSphere Integration Developer converts the source artifacts automatically. • Ensures interoperability and flexibility as part of your service oriented architecture (SOA) through adoption of popular standards, such as BPEL, Web services, JMS, XML, and many more. • Orchestrates the assets of your business to form highly optimized and effective processes to meet your business goals, whether you need to automate processes in the factory, process claims and financial payments, execute an efficient supply chain, or ensure compliance with the latest industry regulations. • Uses WebSphere Integration Developer, providing a developer experience that is second to none. One tool, one set of skills enables you to orchestrate processes, construct mediations between services, and truly integrate the capabilities previously locked away in your packaged business applications. • Contains WebSphere Enterprise Service Bus, which mediates disparate resources, maximizing reuse of your assets wherever they are — irrespective of vendor, platform, or whether they are home-grown or packaged applications. • No special deployment procedures required in most cases, thereby utilizing the already existing WebSphere Application Server Administrator skill sets as the deployment procedures are almost the same. • Native support in the WebSphere Process Server runtime of any WebSphere Interchange Server API's, thereby preserving your existing code if considering migrating from WebSphere Interchange Server. • Support for the current WebSphere Business Integration Adapter technology so that existing adapters will be compatible with the WebSphere Process Server. 	<ul style="list-style-type: none"> • Existing Interchange Server customers, depending on the licensing contract, are automatically entitled to WebSphere Process Server. However for new customers it might be a significant investment. • Need to evaluate if the existing hardware infrastructure supports the new Process Server environment. • Depending upon the scenarios whether Data Integration or Process Integration, it might be an overhead to implement Process Server unless the business solution requires complex work flow processes involving human interaction, on the fly business rules management and so on. • Relatively new product from IBM is in its infancy stages and requires constant releases from IBM for any patches thereby increasing the development and testing time significantly. • Certain out of the box features available in WebSphere Interchange Server are not directly available in WebSphere Process Server and could be implemented using workarounds. • No support for Hot Deployment/Dynamic Update and the deployment procedure is not very efficient and could be time consuming because if multiple modules are dependent upon a library module, then all the modules need to re-deployed if a single artifact is changed in the library module even if it is not referenced in all the modules. This is current limitation and needs a better way from IBM to address this issue.

Three Step Process 1-2-3... Can it really be that easy? Please see Appendix "B"

1. **Assess:** Evaluate what interfaces you have. You probably already have a detailed Inventory/
2. **Analyze:** Now based on what you have, figure out what interfaces you have that are pure data integration, with out any workflow or any complicated business logic in it. Also, figure out what interfaces have complex business logic and some workflow in it. Determine the differences in process integration scenarios and data integration scenarios.
3. **Evaluate Options:** Talk to IBM or a business partner like ProSoft who has specialist for ICS to WPS migrations. Our expertise can mitigate many of the risks. We work closely with Lab Services and request their input and guidance on your behalf.

*Because WebSphere Process Server is built on WebSphere Application Server, customers are able to utilize WebSphere Application Server functionality through their WebSphere Process Server license. In addition, WebSphere Process Server includes additional functions not supplied with WebSphere Application Server. WebSphere Process Server appeals to many WebSphere Application Server customers who are looking to integrate applications running in their enterprise, but do not want to implement the integration required to link the applications by coding bespoke application logic either within the application programs themselves or by coding additional adapter logic. WebSphere Process Server allows the integration of applications with minimal programming, with the integration logic being performed on the data "in flight" between applications.

Common Mistakes and Challenges

- Doing nothing
- Lack of planning
- Dedicating Resources
- Coordination with other teams, testing, and production support knowledge transfer, etc...
- Budget
- Creating a business case

Things to keep in mind when considering migrating from ICS to WPS

When migrating from ICS to WPS, the following need to be taken into consideration. While this list is not complete in providing all the features offered by migration toolset, the main intent is to outline only the things to keep in mind if you are going to follow the migration path. For a most comprehensive list, refer to APPENDIX A.

The success of migration depends on the existing system documentation available. Be sure to capture the integration architecture and design, including the functional design and quality of service requirements, the interdependencies of artifacts shared across projects, and also the design decisions that were made during the deployment. This will assist in system analysis during migration and will minimize any rework efforts.

- Post-migration redesign is recommended as there is a significant functional parity between WebSphere Interchange Server and the current version of WebSphere Process Server.
- No out of the box support is available for existing Crossworlds customers in terms of migrating to WebSphere Process Server as the artifacts are first needed to be converted to 4.2.2 or 4.2.3 or 4.3 before proceeding with the migration.

- The migration tools do not handle Benchmark artifacts in the WebSphere Interchange Server.
- Although, WebSphere Process Server and Integration Developer provides in built support for most of the WebSphere Interchange Server API's, these API's are provided to support migrated artifacts until they can be modified to use new Process Server equivalent API's as the Interchange Server API's are all deprecated and will be removed in the future release.
- If planning to consider legacy WBI Adapters to connect to WebSphere Process Server, then there will be an extra overhead of converting from WebSphere Interchange Server XML to WebSphere Process Server DataObject during runtime. Also care must be taken for certain default values "CxBlank" and "CxIgnore" set in the Interchange Server XML, as they will be interpreted differently in the Process Server.
- Proper evaluation of the assigned data types for Business Objects in the WebSphere Interchange Server is required as in this case the data types are not strongly typed, so there is a possibility of specifying string data type for non-string data attributes. But whereas in WebSphere Process Server, the Business objects within WebSphere Process Server which are based on Service Data Objects (SDOs) utilize data attributes that are strongly typed. So to avoid issues in WebSphere Process Server, be explicit in the specification of data types.
- Common custom code utility libraries for use across WebSphere Interchange Server artifacts need to be carefully evaluated as these might be rendered incompatible in Process Server environment. Thinking in terms of implementing the custom code logic in EJB's deployed on WebSphere Application Server and exposing them as webservices could be one of the recommended approaches.
- For relationships, remember that while relationship definitions will be able to be migrated for use in WebSphere Process Server, the relationship table schema and instance data may be reused by WebSphere Process Server, and also shared concurrently between WebSphere Interchange Server and WebSphere Process Server.
- The support for CORBA IDL interface API's is removed from WebSphere Process Server and so any existing clients such as Access Framework clients needs to be redesigned with another alternative.

APPENDIX A

This section defines a set of best practices for migrating from WebSphere Interchange Server to WebSphere Process Server 6.0 to ensure a smooth transition.

Environment

These guidelines/limitations are applicable for only the following environment and may not be applicable as the issues might be resolved in the latest release V6.0.2.2. For an up to date release, please consult one of our WebSphere consultants at ProSoft.

Operating system(s): AIX, HP-UX, Linux, Solaris, Windows

Software version: WPS 6.0, 6.0.1

Business Objects (BO)

Don't

1. Don't use any spaces in Business Object attribute names.
2. Don't depend on preserved order of entries in an array.
3. Don't rely on array indexing in maps and relationships when referencing into a Business Object.

Business Object Schema

Do

1. Only use Business Object Designer to manipulate business object schemas instead of doing manual modifications.
2. Use explicit specifications of data-types for attributes since in the future all attributes will be strongly typed.

Don't

1. Don't use zero-length attributes (to signify a length of 255).
Instead, use exactly 255 for attribute length if so desired.
2. Don't use 255-length attributes when less is actually required.
Instead, use the exact attribute length required.
3. Don't create Artifacts with ':' in the attribute name. This creates an invalid XSD. XSDs must be valid for WBI Business Objects.
Instead rename to something else before migration. NCName rules apply.

Business Object API

Do

1. Follow currently documented Business Object API only.

Collaboration Templates (CWT)

Do

1. Only have a single triggering port for Collaborations.
2. When variables are declared in java snippets, give them a default value. Such as null for objects.

Don't

1. Don't use Java *Static* in Java Snippets
2. Don't use Java *final/transient/native* qualifiers in Java snippets
3. Don't have multiple triggering ports for Collaborations
4. Don't use 'this' when referencing cwt variables.
Instead of this.inputBusObj use inputBusObj
5. The messageRecipient field in a collaboration is not migrated and not supported for this release.

Modeling

Do

1. Use Process Designer to manipulate collaboration definitions instead of doing manual modifications.
2. Utilize Activity Editor wherever possible.

Variable Scoping

Don't

1. Don't use static variables, since these will not be supported in the future.
Instead, use non-static variables or collaboration properties.
2. Don't use scenario-scoped variables as much as possible, since these will not be supported in future.
Instead, use class-level variables.

APIs

Do

1. Follow currently documented Collaboration APIs only
2. Ensure that explicit connection release calls and explicit transaction bracketing (i.e. explicit commits & explicit rollbacks) are used for User Defined Database Connection Pools API. Reliance on container-managed implicit connection clean-up and implicit transaction bracketing safeguards may no longer be possible in the future.

Don't

1. Don't keep user-defined connection pool transactions active in Java snippet code across process node boundaries for non-LLBP (Long Lived Business Process) collaborations.

Instead, ensure that all such transactions are either committed or rolled back before the end of the node

Fault Handling

1. We only work with single exceptions, not multiple exception types from the same decision node.
2. currentException/currentExceptionMessage are not set.

Break Node

1. BreakNode is empty. Needs code set to make it actually break.

Correlation Sets

1. Not migrated. They need to be setup by hand.

Variables set in Component

1. Iterator does not setup loop variables. It needs to be done manually

Multiple Templates for Multiple Triggering Ports

1. We only do singles for now.

Java

Don't

1. Don't use final/transient/native, they won't be honored. They will cause errors unless java warn is on.

Iterator

1. Conditions for iteration are not migrated in this release, post migration you must go into the bpel while loops and set a condition for them.

Collaboration Object (CWC)

1. If the CWC files in your jar are not complete (do not list all the verbs for each Business Object and their triggering scenarios), try getting the jar from the service manager export instead of reposCopy.

Maps

General

Do

1. Specify all maps in Con files (no implied maps).
2. When doing a "round trip" to/from an adapter, map to the same GBO type on the return that was used on the send, i.e. if GBO type 1 is mapped to ASBO type 1 on the way to the adapter, then ASBO type 1 should be mapped back to GBO type 1 on the return; not to a different GBO type.
3. Use a submap to reference child Business Objects.
4. SET values should be constants instead of java code.

Due to ICS quirks, Java code still runs in ICS but will have problems when migrated. Below is the workaround:

Premigration:

- In the CWM file, change the set value from "xml version=" + "1.0" + " encoding=" + "UTF-8" to "xml version=1.0 encoding=UTF-8" or...

PostMigration

- In the Map file, change:
<map:propertyMap executionOrder="3">
<map:set value="xml version="+1.0+" encoding="+UTF-8" >
<map:output businessObjectVariableRef="WMQ_ContactOutput0" property="XMLDeclaration" />
</map:set>
</map:propertyMap>
to:
<map:propertyMap executionOrder="3">
<map:set value="xml version=1.0 encoding=UTF-8" >
<map:output businessObjectVariableRef="WMQ_ContactOutput0" property="XMLDeclaration" />
</map:set>
</map:propertyMap>

Don't

1. Don't use Java *Static* in Java Snippets
2. Don't use Java final/transient/native qualifiers in Java snippets
3. Don't depend on preserved ordering of the entries in an array. That is, don't use array indexing.

Modeling

Do

1. Only use Map designer to manipulate map definitions instead of doing manual modifications.
2. Utilize Activity Editor wherever possible

Variable Scoping

Don't

1. Don't use static variables, since these will not be supported in the future.

Instead, use non-static variables or map properties.

APIs

Do

1. Follow currently documented Map APIs only
2. Ensure that implicit connection release calls and implicit transaction bracketing (i.e. implicit commits & implicit rollbacks) are used for User Defined Database Connection Pools API. Reliance on container-managed implicit connection clean-up and implicit transaction bracketing safeguards may no longer be possible in the future.

Don't

1. Don't keep transactions active in Java snippet code across transformation node boundaries.

Instead, ensure that all such transactions are either committed or rolled back before the end of the node

Binding

Do

1. Ensure that all maps are explicitly bound. Only those maps that are explicitly bound are recognized.

Mediations

Do

1. Sync mediation steps are always created, so either the user needs to create return maps before migration or remove the unnecessary mediations after migration

Matching Maps

Do

1. If you define an ASBO 2 GBO map, define the GBO 2 ASBO map as well. If not you will need to edit post migration

Java

Don't

1. Don't use final/transient/native, they won't be honored. They will cause errors unless java warn is on.

Java Snippet Code

Do

1. Ensure that Java snippet code will be compatible to run in an EJB container in a J2EE environment
2. Adhere to development responsibilities as provided in the EJB Specification Version 2.1, Chapter 25 Runtime Environment (<http://java.sun.com/products/ejb/docs.html>).

Don't

1. Don't spawn threads or use thread synchronization primitives.
If you must, these will need to be converted to use Asynchronous Beans when you migrate.
2. Don't do any disk I/O using java.io.*
Use JDBC to store any data
3. Don't perform any functions that may be reserved for an EJB container such as socket I/O, classloading, loading native libraries, etc.
If you must, these snippets would need manual conversion to use EJB container functions when you migrate.

Adapters

Adapter Configs

Do

1. Specify all maps in Con files (no implied maps)

C++ Adapters

Don't

1. Don't use the C++ Adapter Development Kit (ADK) any longer.
Instead, it is recommended, if possible, that all current C++ adapters should migrate to use the Java CDK. If this migration is not possible, then samples that utilize the JNI (Java Native Interface) to make calls to the C++ EIS APIs under the purview of the Java CDK will be made available.

Threading Models

Don't

1. Don't use the MAIN_SINGLE_THREADED threading model, as this will not be supportable in the future.
Instead, explore if the underlying EIS has a new re-entrant API that doesn't require the MAIN_SINGLE_THREADED option.

APIs

Do

1. Follow currently documented Java Adapter Development Kit (JADK) APIs only

Relationships

Don't

1. Relationship coexistence is not supported in this release.

Modeling & Schema

Do

1. Use Relationship designer to manipulate relationship definition instead of doing manual modifications.

Don't

1. Never alter relationship role table schema manually or using SQL scripts.
Instead, always use Relationship Designer for this.

APIs

Do

1. Follow currently documented Relationship APIs only

Don't

2. The heritage api call for maintainIdentityRelationship is not functional for context SERVICE_CALL_RESPONSE

Instance Manipulation

Do

1. Use Relationship Instance Manager to manipulate relationship instances instead of manually or using SQL scripts.

Access Framework Clients

API

Do

1. Non-EJB clients, use WebServices adapter for now for all access interface clients. We will provide seamless migration between WebServices adapter clients and the new Access Framework since both are WSDL based.
2. Clients that have access to EJBs should try and use the Access framework supplied EJB if possible.

Don't

1. Don't develop any new clients adopting the CORBA IDL interface APIs.
Instead, use the Web Services adapter or Access Framework EJB as described above.

Utility Libraries

Don't

1. Avoid creating new homegrown content utility libraries, as these will be difficult (if not impossible) to migrate. No migratability guarantees can be made. Onus of migration of such utilities will be on Client Services or Customer Engagement Team, whoever developed it.

Instead, try and use available APIs as much as possible or encapsulate the utility in an EJB.

Runtime Transaction Log Data

Runtime Data Manipulation

Do

1. All runtime data should be manipulated using the Flow Manager tool, which includes provisions for resubmitting unresolved flows and repair in-doubt transactional collaborations.

Don't

1. Avoid any direct manipulation of runtime data (either manual, via SQL scripts or using undocumented APIs).

Instead, use the Flow Manager as recommended above.

Interoperability with other WBI brokers such as WBI-MB (a.k.a. WMQI) and WMQWF

Do

1. To interoperate between WICS and WBI-MB (Message Broker) (a.k.a. WMQI), use the WBI-MB adapter.

2. To interoperate between WICS and WebSphere MQ Work Flow broker, use the WMQWF adapter.

Configuration Files (CFG)

Asynchronous Response Input Wiring

1. User must wire asynchronous response input.

Sync input Wiring

1. User must wire sync input somehow

Setup Publish on multiple Imports

1. First target is chosen. User must setup publish through admin console.

Collaboratoin Objects (CWC)

1. We only wire to the first cwc found, the rest need to be wired manually in the admin console.

XML2Java – user templates

1. Uses standard templates for now.

ICS 4.3 Security

1. Not done in this release.

Tables below list specific API's that are not currently supported but will be in the future:

Base Collaboration	
Collaboration	dropFailedEvent(String, String, int)
	dynamicSend(String, String, String, BusObj, String)
	queryFailedEvents(EventqueryDef[], String, String, int, int): FailedEventInfo[]
	resubmitFailedEvent(String, String, int, int, int)
	saveFailedEvent(BusObj)
	sendEmail(String, String, Vector)

EventKeyAttrDef

EventManagement	EventKeyAttrDef(String, String)
CxCommon	public String keyName
	public String keyValue

EventQueryDef	
EventManagement	EventQueryDef(String, String, String, String, int)
CxCommon	public String nameConnector
	public String nameCollaboration
	public String nameBusObj
	public String verb
	public int ownerType

FailedEventInfo	
EventManagement	FailedEventInfo(String x6, int, EventKeyAttrDef[], int, int, String, String, int)
CxCommon	public String nameOwner
	public String nameConnector
	public String nameBusObj
	public String nameVerb
	public String strTime
	public String strMessage
	public int wiplIndex
	public EventKeyAttrDef[] strbusObjKeys
	public int nKeys
	public int eventStatus

	public String expirationTime
	public String scenarioName
	public int scenarioState

Syntactical errors

1. CWT properties could have any name, BPEL properties can not have special characters or start with numbers. So pre/post migration the user may need to change the names of these properties to avoid syntactical errors.

DB Websphere Variables

1. The jdbc provider and datasources created by ics migration may need variables set in websphere to work.

Legacy WBI Adapters

1. async in response does not work in this release
2. sync in does not work in this release

APPENDIX B

Tricky Situations to consider- Please evaluate your environment and Architecture specifically in the areas defined below.

Migration from WebSphere InterChange Server to Websphere Process Server isn't as simple as 1-2-3 as we portrayed earlier. ProSoft is here to support your specific solution.

To enable migrated applications to function in WebSphere® Process Server consistently with their intended function in WebSphere InterChange Server, special attention is required in some areas due to differences between the architectures of WebSphere Process Server and WebSphere InterChange Server.

Compensation levels

Since there is no direct mapping of the levels of compensation between WebSphere InterChange Server collaborations and WebSphere Process Server BPEL, the compensation level specified in the WebSphere InterChange Server Collaboration is ignored and the default BPEL compensation level will be used in the migrated application. You should understand BPEL compensation and adjust your migrated applications accordingly to get the desired functionality.

InstallAdministrativeObjects.py script

For WebSphere InterChange Server artifacts that have no corresponding artifact in WebSphere Process Server, a Jython® script is generated during migration. The script can be run with the wsadmin command to create WebSphere Process Server configuration definitions corresponding to the original WebSphere InterChange Server artifacts. The WebSphere InterChange Server artifacts that are handled in this manner are DBConnection pools, relationship databases, and scheduler entries. The Jython script that is generated is called InstallAdministrativeObjects.py, and a copy of it will be included wherever the shared artifacts are included. That is, the InstallAdministrativeObjects.py script is included in every jar file created by the ReposMigrate command, and it is put in the library project specified during import in WebSphere Integration Developer. An InstallAdministrativeObjects.py script is always generated even if there are no artifacts that require it. This script can be modified to add or delete entries before using wsadmin to execute it.

Note the following:

- The InstallAdministrativeObjects.py script will use default JDBC provider templates to create JDBC providers if an appropriate JDBC provider is not already defined. A side effect of using these default JDBC provider templates is that an empty, sample datasource definition is created. This sample datasource is not used. Delete it to prevent exceptions that might occur during server start-up due to the fact that the sample datasource does not specify all of the information required for a datasource.
- To simulate the WebSphere InterChange Server environment, that is, an environment in which resources are defined once for the WebSphere InterChange Server system, the InstallAdministrativeObjects.py script defines resources at the cell scope in WebSphere Process Server. WebSphere variables are predefined in the WebSphere Process Server system for use by JDBC providers created from the default JDBC provider templates. However, these variables are defined at the node scope since they can be customized for each node. Because of this scoping discrepancy, you will need to do one of the following: define WebSphere Variables needed by the created JDBC providers at the cell scope, or move the JDBC provider to the node scope after running the InstallAdministrativeObjects.py script. Use the administrative console to examine the JDBC providers that are generated to determine which WebSphere variables are needed. From the administrative console, select Environment > WebSphere Variables to create any required variables. (For more information, see "Configuring WebSphere Variables" at the WebSphere Application Server Network Deployment information center.

Access Enterprise JavaBean (EJB) support

WebSphere InterChange Server supports the triggering of collaborations by client code via a J2EE EJB (Enterprise JavaBeans™) protocol. Support for this method of triggering collaborations is referred to as "Access EJB" or "Access for EJB" support. For backward compatibility, WebSphere Process Server provides deprecated support for Access

EJB. The Access EJB support assumes that the SCA BPEL modules to be invoked were generated by the WebSphere InterChange Server migration tools described in this documentation. The mapping from the collaboration name and port name (that is, the input parameters for the Access EJB) to the SCA module name, interfaces and business object types assume the conventions used by the migration tools. The Access EJB support in WebSphere Process Server is delivered in the AccessEJB.zip project interchange file. It can be found in the install_root/HeritageAPI directory. The AccessEJB support consists of an EJB (Access EJB) which references an SCA module project (DynamicRouting) that invokes the SCA BPEL module. This SCA BPEL module is the migrated version of the collaboration that was invoked in WebSphere InterChange Server. The DynamicRouting module uses a selector component to select the correct SCA target based on the collaboration name and port name passed to the Access EJB. To enable Access EJB support in WebSphere Process Server, do the following:

1. Import the WebSphere InterChange Server repository containing the collaboration that is the target of the Access EJB invocation into WebSphere Integration Developer.
2. Import the AccessEJB.zip project interchange file into WebSphere Integration Developer.
3. Open the DynamicRouting project and update the selector table to include the migrated module that is to be invoked via the Access EJB.
4. Go to the migrated project containing the BPEL component to be invoked via the Access EJB, and drag the export that references the BPEL module over to the DynamicRouting project.
5. Repeat steps 3 and 4 for each BPEL module that is to be accessible via the Access EJB.
6. Build the project and deploy it to the WebSphere Process Server server.
7. Ensure that any required data handlers are provided in the runtime classpath of the WebSphere Process Server server.
8. To enable your Access client to use WebSphere Process Server, ensure that it points to the WebSphere Process Server server and uses the JNDI name com/crossworlds/access/business/cwsession/CwSession when looking up the Access EJB.

DynamicSend API

In WebSphere InterChange Server, the DynamicSend API can be used to directly invoke one collaboration from another. The collaboration to be invoked does not have to be predetermined, instead, it can be determined dynamically at runtime. The support for the DynamicSend API in WebSphere Process Server uses the DynamicRouting project described in "Access Enterprise JavaBean (EJB) support." Follow the instructions in that section to enable the DynamicSend API to be able to invoke the specified BPEL modules.

Security

Methods are available for securing a business integration system with WebSphere Process Server in ways equivalent to the security which could be achieved with WebSphere InterChange Server.

Event sequencing

Methods are available for sequencing events with WebSphere Process Server in ways similar to the way you could with WebSphere InterChange Server.

Failed events

Methods for handling failed events in WebSphere Process Server are described in article(s) that you might find helpful on the IBM developerWorks web site. Look in the "Technical Library" at <http://www.ibm.com/developerworks>.

Supporting Links:

<http://www-306.ibm.com/software/integration/wps/> - Main WebSphere Process Server Page

http://www.ibm.com/developerworks/blogs/page/woolf?entry=websphere_process_server_a_russian Bobby Woolf

<http://www-306.ibm.com/software/integration/wid/> - WID WebSphere Integration Developer