Using PeopleTools for Seamless Integration with Oracle's SOA Suite and BPEL

A Solbourne White Paper
April 2008

Solbourne

Using PeopleTools for Seamless Integration with Oracle's SOA Suite and BPEL

Executive Overview

As the heart of Oracle's SOA Suite, Oracle's BPEL PM provides excellent integration opportunities for PeopleSoft applications. This presentation will provide an introduction to using Oracle's BPEL Process Manager with PeopleTools for real-time SOA integration. We'll discuss leading practices for using PeopleTools' Integration Broker, and look at ways to maximize your investment by creating BPEL processes that are designed to consume/provide web services for PeopleTools.

Introduction

The PeopleSoft suite of applications is known for its elegant interface and cutting edge architecture. From a developer's perspective, PeopleTools follows suit as an integrated set of tools that helps extend and customize PeopleSoft. One of the powerful components of PeopleTools is the Integration Broker. Integration Broker allows developers to create web services from standard PeopleSoft business processes, and consume web services from external systems, easily and quickly.

This paper will provide an overview of the Integration Broker, as well as some key leading practices for developing interfaces which involve Oracle's BPEL Process Manager. The whitepaper will start with PeopleSoft and work our way into the Middleware arena. We'll also touch on some high-level considerations as these interfaces extend into Oracle's E-Business Suite.

Assumptions

This paper assumes PeopleSoft will be an integral part of the business environment where any data integrations are planned. A basic understanding of what SOA (Service Oriented Architecture) is and how Oracle's SOA Suite fits into an ERP system will also be helpful. PeopleSoft and E-Business Suite Development concepts will also be covered at a high level.

Overview

Traditional systems integration has been performed with the following overarching concepts:

1. Point-to-Point architecture
    a. Data are moved from one point to the next, usually using a file to hold data en route
    b. The source and target systems handle all data transformation, validation, and processing
    c. Moving data is the primary concern
2. Data are processed in batches
    a. Large volumes of data are processed on a regularly scheduled routine, generally during non-business hours. The overhead of creating and loading file-based data lends itself to batch processing.
    b. Reports handle the capture of errors and erred data are resubmitted as part of the subsequent interface process

As systems have evolved and both business needs and user expectations have changed, the face of integrations has also changed. Due largely to its origin in W3C (World Wide Web Consortium) standards and open source development, Service Oriented Architecture has been placed at the forefront of technologies to handle these changing expectations. With a SOA platform as the middle ground for executing integrations, the following characteristics become much easier to achieve:

1. Integrations focus more on Process, less on data
    a. Integrating additional steps that more logically bring two integrated systems together can be done more easily
        i. An invoice moving from one system to the next can be approved before it is entered in the target system
    b. Data can be validated, and processed in the middle ground between systems, leaving the source and target systems to do what they do best, rather than forcing integration logic into them.
2. Integrations can *more easily* happen in real time
    a. Smaller volumes of data are processed as business happens
    b. Error handling can be done in real time, with less impact on the source system – corrections to data are done in the middleware arena
    c. SOA platform is naturally better at real time, although it can do batch processing as well.

PeopleSoft was one of the first ERP systems to develop tools to handle interaction with web services, the backbone of Service Oriented Architecture. Oracle's SOA Suite has emerged as the pre-eminent platform for bridging the gap between systems like PeopleSoft and third-party applications of all kinds. Together, they create tremendous capabilities for meeting and exceeding ERP integration expectations.

Integration Broker Overview

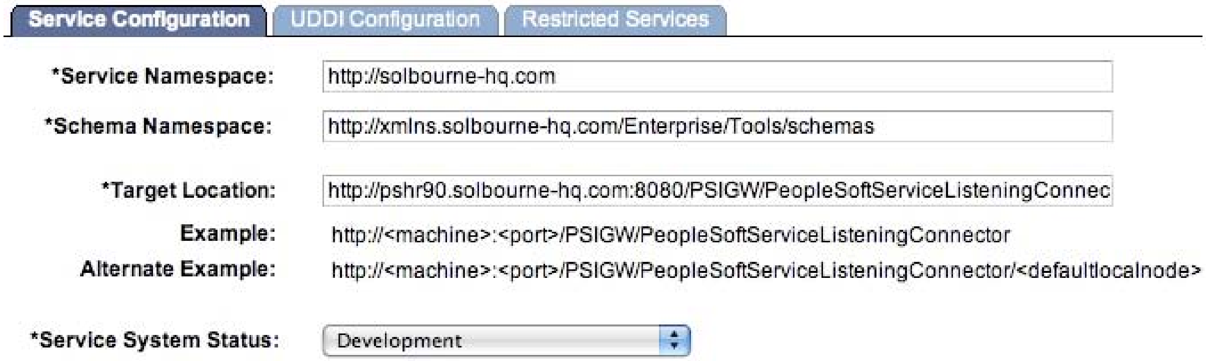
The Integration Broker facilitates XML messaging for PeopleSoft. The Integration Broker provides tools for internal messages, consuming external web services, providing web services for external and 3rd party systems. The Integration Broker was introduced with PeopleTools 8.4 and updated in PeopleTools 8.48 to better handle SOA-based services. This whitepaper will show how to use the Integration Broker on PeopleTools 8.48 and above.

To configure the Integration Broker for basic XML operations, consult PeopleBooks or reference Customer Connection. This whitepaper will discuss specific configurations for consuming and provided SOA-based services.

Creating Web Services with the Integration Broker

With PeopleTools 8.48, PeopleSoft introduced an improved Integration Broker that supports easier tools to consume and provide SOA-based services. The Service Configuration component allows you to define namespaces, UDDI servers, and control access to specific services. Ensure the Target Location users the "PeopleSoftServiceListeningConnector". This location does not use the same listening servlet as the Integration Gateway Listener.

PeopleTools > Integration Broker > Configuration > Service Configuration



*Service Namespace: http://solbourne-hq.com

*Schema Namespace: http://xmlns.solbourne-hq.com/Enterprise/Tools/schemas

*Target Location: http://pshr90.solbourne-hq.com:8080/PSIGW/PeopleSoftServiceListeningConnec

Example: http://<machine>:<port>/PSIGW/PeopleSoftServiceListeningConnector

Alternate Example: http://<machine>:<port>/PSIGW/PeopleSoftServiceListeningConnector/<defaultlocalnode>

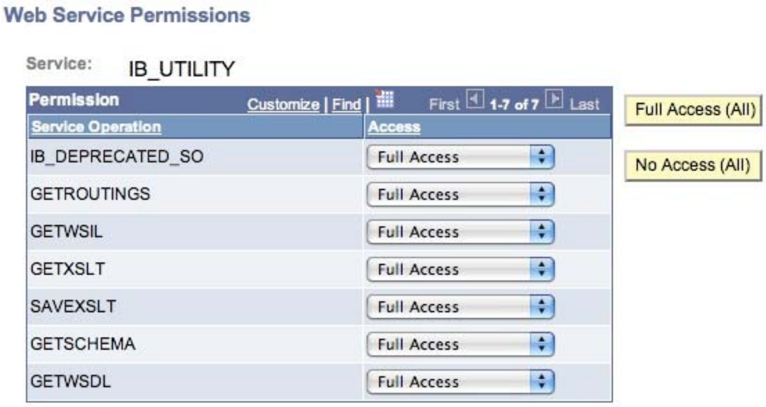
*Service System Status: Development

Web Service Security

The new PeopleSoftServiceListeningConnector servlet uses and web service PeopleCode uses new Service Operations and Web Libraries. There are special Service Operations that must be assigned to a Permission List for integrations using PeopleSoft Generated WSDL and SOAP objects.

PeopleTools -> Security -> Permissions & Roles -> Permission Lists

Select a Permission List that is assigned to the user who starts the application server. Add security to the Web Service IB_UTILITY and assign Full Access to each Service Operations. These operations are used in the WSDL repository, when routing incoming and outgoing messages and for transforming message schemas.



Web Service Permissions

Service: IB_UTILITY

Permission

Service Operation	Access
IB_DEPRECATED_SO	Full Access
GETROUTINGS	Full Access
GETWSIL	Full Access
GETXSLT	Full Access
SAVEXSLT	Full Access
GETSCHEMA	Full Access
GETWSDL	Full Access

Using Web Service Wizards

There are two wizards that will help setup the service operations and routings for web service integrations.

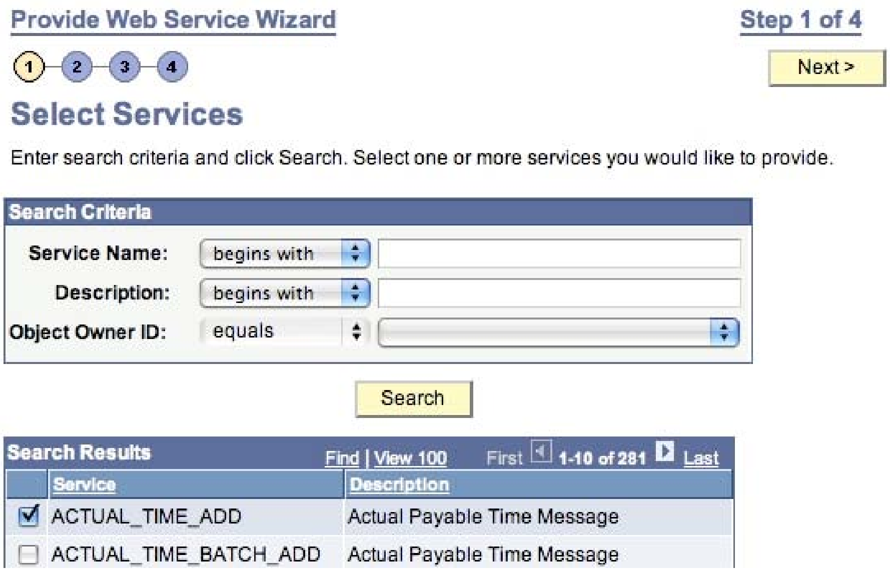
- Provide a Web Service (Expose PeopleSoft so it is available for another system)
- Consume a Web Service (Send data to an external web service)

Provide a Web Service

The Provide Web Service wizard simplifies the process of exposing PeopleSoft services to external systems. The wizard creates the necessary WSDL file, adds the WSDL file to the WSDL repository or UDDI server, and creates the correct routings for the service.

PeopleTools -> Integration Broker -> Web Services -> Provide Web Service

The Provide Web Service wizard will let you select Service Operations to expose. You can choose to expose all, some or just one operation. The wizard will use this selection to create the appropriate sections in the WSDL file.



Provide Web Service Wizard

Step 1 of 4

Select Services

Enter search criteria and click Search. Select one or more services you would like to provide.

Search Criteria

Service Name: begins with

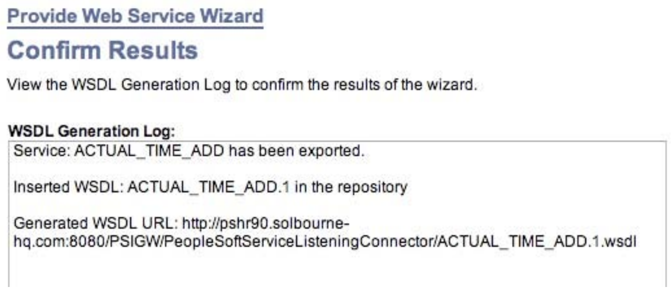
Description: begins with

Object Owner ID: equals

Search Results

	Service	Description
☑	ACTUAL_TIME_ADD	Actual Payable Time Message
□	ACTUAL_TIME_BATCH_ADD	Actual Payable Time Message

Once you finish the wizard, a confirmation page displays the URL for the new service’s WSDL location. To verify the WSDL location, copy the WSDL URL in the confirmation box and paste the URL into the address box of you web browser. The location will display the new WSDL file.



Provide Web Service Wizard

Confirm Results

View the WSDL Generation Log to confirm the results of the wizard.

WSDL Generation Log:

```
Service: ACTUAL_TIME_ADD has been exported.

Inserted WSDL: ACTUAL_TIME_ADD.1 in the repository

Generated WSDL URL: http://pshr90.solbourne-
hq.com:8080/PSIGW/PeopleSoftServiceListeningConnector/ACTUAL_TIME_ADD.1.wsdl
```

After the WSDL file has been verified, the application server user must be granted security to the new web service. Open a Permission List assigned to the user who starts the application server and grant Full Access to the new web service.

If you are working with an external system, PeopleTools can generate SOAP message templates for use with this new web service. This page is available at

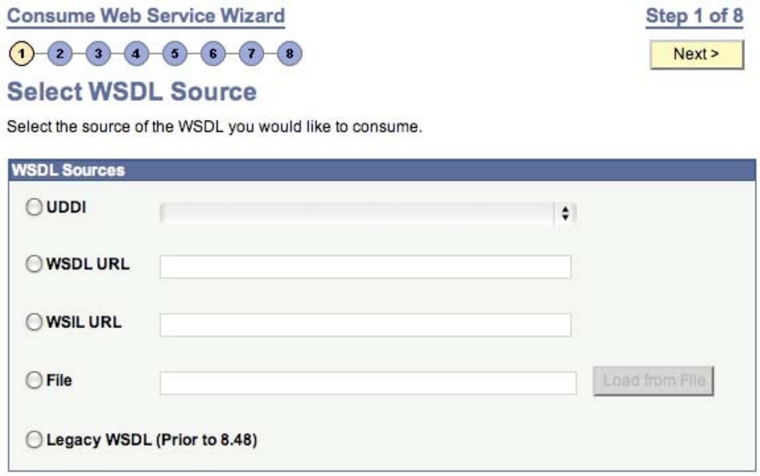
PeopleTools >Integration Broker > Service Utilities > Generate SOAP Template

Consume a Web Service

The Consume Web Service wizard simplifies the process of configuring PeopleSoft to use an external web service. The wizard will need the URL for the web service's WSDL file, or flat file copy of the WSDL. The Consume Web Service wizard creates new message objects (schema), new routings and target location according to the web service WSDL.

PeopleTools > Integration Broker > Web Services > Consume Web Service

The first page of the wizard asks for the location of the external web service's WSDL file. You can select a service from a UDDI server (if configured), WSDL or WSIL URL or a flat file. If you are interfacing with a PeopleSoft system running PeopleTools 8.47 and earlier, there is a different option to import the legacy formatted WSDL file.



Select WSDL Source

Select the source of the WSDL you would like to consume.

WSDL Sources

○ UDDI

○ WSDL URL

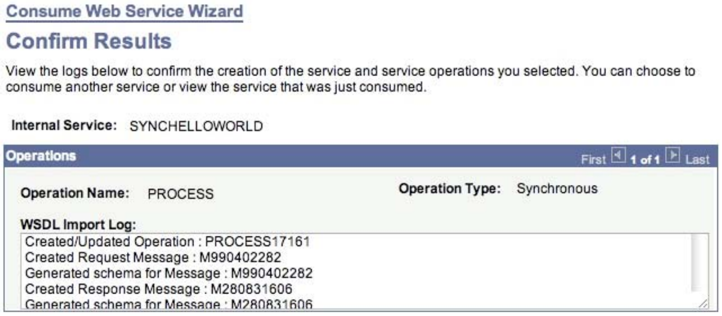
○ WSIL URL

○ File

○ Legacy WSDL (Prior to 8.48)

Once you have entered or loaded the file, the system will check the WSDL for available services and display them. The wizard will create a new Service and Service Operation based on information in the WSDL. The names of the Service and Service Operation will be taken from the WSDL. Next, the wizard will create the Request and Response Message schemas for the external web service. According to PeopleBooks, the naming convention for auto-generated messages is an "M" followed by a unique string of six numbers. The last step is to select a destination node. The node contains the external system information (i.e. where to send the XML data). By default, select the WSDL_NODE for all BPEL-based integrations.

The confirmation page will list the new message schema objects and the new Service and Service Operations.



Consume Web Service Wizard

Confirm Results

View the logs below to confirm the creation of the service and service operations you selected. You can choose to consume another service or view the service that was just consumed.

Internal Service: SYNCHELLOWORLD

Operation Name: PROCESS

Operation Type: Synchronous

WSDL Import Log:

```
Created/Updated Operation : PROCESS17161
Created Request Message : M990402282
Generated schema for Message : M990402282
Created Response Message : M280831606
Generated schema for Message : M280831606
```

To use the new web service, a developer will need to add PeopleCode to populate the new message and publish the data. Reference the PeopleCode API PeopleBook, or re-use PeopleSoft delivered code if the message structure is similar to an existing message.

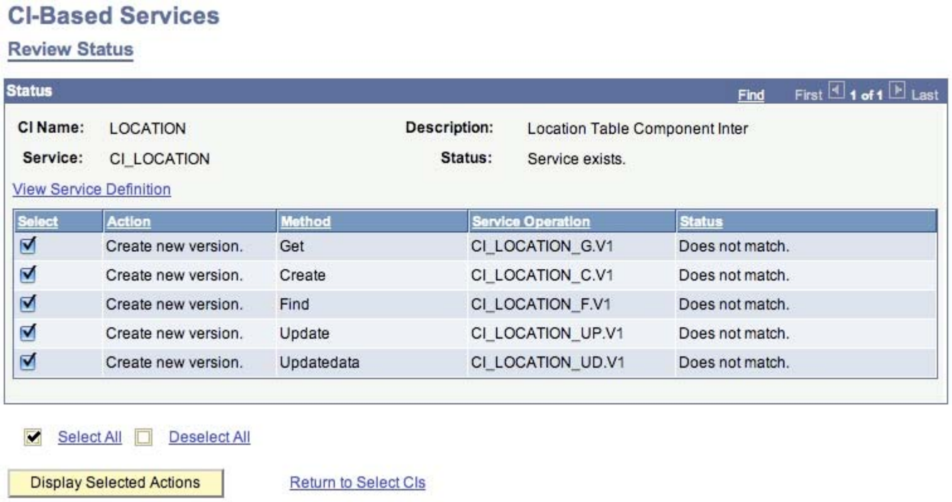
Also, the end user who will be executing the PeopleCode that publishes data to this web service must have access to the new web service. Ensure the appropriate Permission List is updated to prevent the user from experiencing an error.

Component Interface-based Services

Component Interfaces (CI) can be exposed as a web service. The web service will expose the same methods that are available in the CI. A new Service Operation will be created for each CI method.

PeopleTools > Integration Broker > Web Services > CI-Based Services

The first step is to select a CI to be exposed as a service. Click the Review CI Status button to start the exposing the CI as a service. The next page will display the methods that are available from the CI. Select the methods that you want to include in the new service.



CI-Based Services

Review Status

CI Name: LOCATION

Description: Location Table Component Inter

Service: CI_LOCATION

Status: Service exists.

Select	Action	Method	Service Operation	Status
☑	Create new version.	Get	CI_LOCATION_G.V1	Does not match.
☑	Create new version.	Create	CI_LOCATION_C.V1	Does not match.
☑	Create new version.	Find	CI_LOCATION_F.V1	Does not match.
☑	Create new version.	Update	CI_LOCATION_UP.V1	Does not match.
☑	Create new version.	Updatedata	CI_LOCATION_UD.V1	Does not match.

Once the service is created, the CI-Based Services page will show that the new service has been created and that the service operations are also created.

To expose the service for external systems, run the Provide Web Service wizard.

Ensure the application server user has the appropriate security for the new CI-Based service. The application server user will need access to the CI through Permission List security. Also, ensure the Permission List has full access to the SOAPTOCI Service Operation.

Working with the XML Schemas Generated by the Integration Broker to Create BPEL Processes

Schemas are the backbone of SOA/web services development. Schemas both define and validate the message types[1] that are used to pass XML data between systems. When you create the Integration Broker component to expose PeopleSoft logic as a web service, you also create the schema that will be used by any services that consume it.

The catch is that the schema you'll get directly from the Integration Broker is *not quite* ready for use in a BPEL process. The basic structure is great, and offers a very rich and complex data set. However the structure of that schema needs to be massaged to be 100% compatible with the schema requirements of a BPEL process, to create variables based on the elements in that schema.

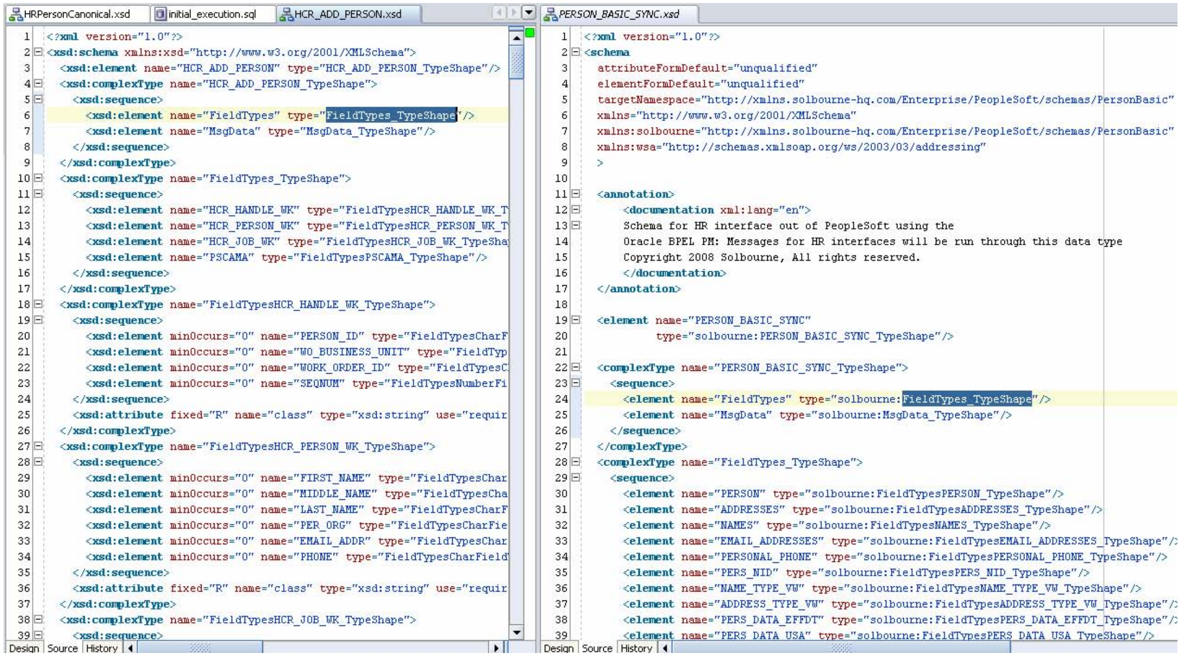
Remember that schemas both define XML and are *themselves* XML. This means that what makes a schema well-formed (validates as XML 1.0 or whatever version is being used) does not necessarily make it *valid* for the system that needs to use it. In the case of the PeopleSoft schemas, the documents are well-formed XML, but do not pass the requirements of being used in BPEL process. Neither system is wrong or right, it's just a difference in requirement.

Thankfully, we can relatively quickly change the schema coming out of PeopleSoft to make it recognizable and usable in a BPEL process. We'll walk through the changes you'll need to make in the schema to get it working. Using JDeveloper as the XML editor, let's take a look at the steps required to prepare a PeopleSoft schema for use in a BPEL process.

This first image shows the file created by PeopleSoft (left side) and end result, a schema ready for use in a BPEL process (right side). The images in this section are small, but by zooming to 200% you can see the files in their natural size. This was done deliberately to preserve the detail for comparison.

[1] Message types are the variables of the SOA world. They can be simple data types (String, integer) or they can be more complex data types similar to record types in PL/SQL, Arrays or Classes in Java, or a Struct in C/C++

To start, we'll focus on the 'header' section, the area that is prior to any data elements being defined. The schema that comes from PeopleSoft leaves out some key components that have been added on the right-hand side.



```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="HCR_ADD_PERSON" type="HCR_ADD_PERSON_TypeShape"/>
  <xsd:complexType name="HCR_ADD_PERSON_TypeShape">
    <xsd:sequence>
      <xsd:element name="FieldTypes" type="FieldTypes_TypeShape"/>
      <xsd:element name="MsgData" type="MsgData_TypeShape"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="FieldTypes_TypeShape">
    <xsd:sequence>
      <xsd:element name="HCR_HANDLE_WK" type="FieldTypesHCR_HANDLE_WK_T
      <xsd:element name="HCR_PERSON_WK" type="FieldTypesHCR_PERSON_WK_T
      <xsd:element name="HCR_JOB_WK" type="FieldTypesHCR_JOB_WK_TypeSha
      <xsd:element name="PSCAMA" type="FieldTypesPSCAMA_TypeShape"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="FieldTypesHCR_HANDLE_WK_TypeShape">
    <xsd:sequence>
      <xsd:element minOccurs="0" name="PERSON_ID" type="FieldTypesCharF
      <xsd:element minOccurs="0" name="WO_BUSINESS_UNIT" type="FieldTyp
      <xsd:element minOccurs="0" name="WORK_ORDER_ID" type="FieldTypesC
      <xsd:element minOccurs="0" name="SEQNUM" type="FieldTypesNumberFi
    </xsd:sequence>
    <xsd:attribute fixed="R" name="class" type="xsd:string" use="requir
  </xsd:complexType>
  <xsd:complexType name="FieldTypesHCR_PERSON_WK_TypeShape">
    <xsd:sequence>
      <xsd:element minOccurs="0" name="FIRST_NAME" type="FieldTypesChar
      <xsd:element minOccurs="0" name="MIDDLE_NAME" type="FieldTypesCha
      <xsd:element minOccurs="0" name="LAST_NAME" type="FieldTypesCharF
      <xsd:element minOccurs="0" name="PER_ORG" type="FieldTypesCharFie
      <xsd:element minOccurs="0" name="EMAIL_ADDR" type="FieldTypesChar
      <xsd:element minOccurs="0" name="PHONE" type="FieldTypesCharField
    </xsd:sequence>
    <xsd:attribute fixed="R" name="class" type="xsd:string" use="requir
  </xsd:complexType>
  <xsd:complexType name="FieldTypesHCR_JOB_WK_TypeShape">
    <xsd:sequence>
```

```
<?xml version="1.0"?>
<schema
  attributeFormDefault="unqualified"
  elementFormDefault="unqualified"
  targetNamespace="http://xmlns.solbourne-hq.com/Enterprise/PeopleSoft/schemas/PersonBasic"
  xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:solbourne="http://xmlns.solbourne-hq.com/Enterprise/PeopleSoft/schemas/PersonBasic"
  xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing"
  >

  <annotation>
      <documentation xml:lang="en">
      Schema for HR interface out of PeopleSoft using the
      Oracle BPEL PM: Messages for HR interfaces will be run through this data type
      Copyright 2008 Solbourne, All rights reserved.
      </documentation>
  </annotation>

  <element name="PERSON_BASIC_SYNC"
           type="solbourne:PERSON_BASIC_SYNC_TypeShape"/>

  <complexType name="PERSON_BASIC_SYNC_TypeShape">
    <sequence>
      <element name="FieldTypes" type="solbourne:FieldTypes_TypeShape"/>
      <element name="MsgData" type="solbourne:MsgData_TypeShape"/>
    </sequence>
  </complexType>
  <complexType name="FieldTypes_TypeShape">
    <sequence>
      <element name="PERSON" type="solbourne:FieldTypesPERSON_TypeShape"/>
      <element name="ADDRESSES" type="solbourne:FieldTypesADDRESSES_TypeShape"/>
      <element name="NAMES" type="solbourne:FieldTypesNAMES_TypeShape"/>
      <element name="EMAIL_ADDRESSES" type="solbourne:FieldTypesEMAIL_ADDRESSES_TypeShape"/
      <element name="PERSONAL_PHONE" type="solbourne:FieldTypesPERSONAL_PHONE_TypeShape"/>
      <element name="PERS_NID" type="solbourne:FieldTypesPERS_NID_TypeShape"/>
      <element name="NAME_TYPE_VW" type="solbourne:FieldTypesNAME_TYPE_VW_TypeShape"/>
      <element name="ADDRESS_TYPE_VW" type="solbourne:FieldTypesADDRESS_TYPE_VW_TypeShape"/
      <element name="PERS_DATA_EFFDT" type="solbourne:FieldTypesPERS_DATA_EFFDT_TypeShape"/
      <element name="PERS_DATA_USA" type="solbourne:FieldTypesPERS_DATA_USA_TypeShape"/>
```

1. The schema element has had several attributes added to it, which include:
    a. attributeFormDefault
    b. elementFormDefault
    c. targetNamespace
    d. Two (2) namespace additions

   Whether the attributeFormDefault or elementFormDefault attributes are set to 'qualified' or 'unqualified' depends on how the schema is generally laid out. For the schemas coming from PeopleSoft, it's generally easier to set them to 'unqualified' as this meshes well with the base structure. The targetNamespace element is important as it is used later by the BPEL process to import the schema so that the data structures can be used to create Messages within the BPEL process.

   The use of namespaces (lines starting with 'xmlns:') is an important way to reference other schemas within a schema and to separate out fields that could be defined within those two differing schemas. In this case, the 'wsa' namespace (a web services addressing namespace put out collaboratively by IBM, BEA, and Microsoft) and the 'solbourne' Namespace simply qualifies that for the purpose of this document, all elements labeled with 'solbourne' will pertain to the data elements in this schema.[2]

2. An annotation has been added to allow for commenting
3. The field names have been change to represent that they will be used as part of the Middleware arena, that is, they will be used in the BPEL process. Although they correspond to the elements in PeopleSoft, we may want to add additional elements later, and establishing a "middleware" naming convention allows us to better compartmentalize this schema.

Moving on to the body of the schema, the next thing to notice is that the 'xsd:' part of every element has been removed. This removes a layer of complication from the schema that is both in tune with how Oracle creates

[2] For more on namespaces, see http://www.w3schools.com/schema/default.asp

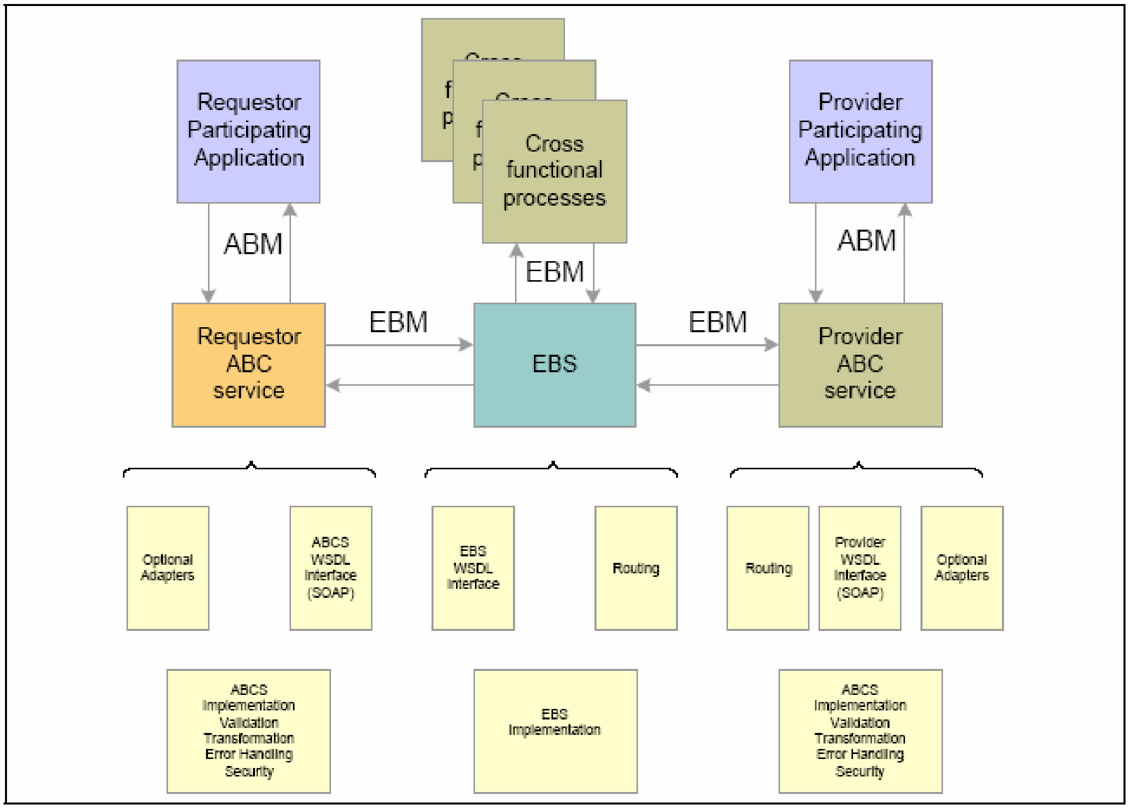
schemas and makes debugging much easier. Finally, the ‘type’ elements have all been prefixed with the ‘solbourne:’ namespace.

In Conclusion, remember that we’re not changing the structure of the data, or even the basic structure of the document. We’re just modifying the syntax and adding extra elements to allow this structure to be used in a BPEL process.

Leading Practices for Creating BPEL Processes that Interact with PeopleSoft

Following Oracle’s lead, Solbourne uses the concepts laid out in the stated direction of Oracle’s Appliation Integration Architecture (AIA) on all Fusion Middleware projects. This architecture provides both leading practices for doing SOA development in general and Oracle’s stated direction for how future Oracle-branded integration components will be designed. Sticking to this as closely as is possible promises both better consistency now within our own projects and better coordination with Oracle as more AIA components are developed. [3]

Enterprise Business Objects (EBO) are the core of this architecture and focus the *data* of an interface. An EBO consists of all elements that comprise a particular business entity, such as a person. A person can be hired, terminated, etc., and all the data elements that a person could possibly have as a result of these operations would come together under the umbrella of a person EBO.


Requestor Participating Application
Cross functional processes
Provider Participating Application
ABM
EBM
ABM
Requestor ABC service
EBM
EBS
EBM
Provider ABC service
Optional Adapters
ABCS WSDL Interface (SOAP)
EBS WSDL Interface
Routing
Routing
Provider WSDL Interface (SOAP)
Optional Adapters
ABCS Implementation Validation Transformation Error Handling Security
EBS Implementation
ABCS Implementation Validation Transformation Error Handling Security

Oracle Application Integration Architecture components

[3] Taken from “Oracle® Application Integration Architecture Foundation 2.0: Concepts and Technologies Guide”, Release 2.0, Part No. E10918-01.

While EBOs are about data, the Application Business Connectors (ABC) are about operations on those data. To compare these components to traditional programming languages, these are the accessor methods of the SOA world. ABCs have one reason to exist and that's to access data and perform operations on it. ABCs are limited, however, to specific operations such as creating a person. The primary five operations [4] that Oracle states should be included for accessing any EBO, and for which we would create these Connectors are:

- Create - to create an object instance
- Update - to update an object instance with only the changes that occurred
- Delete - to delete an object instance
- Query - to retrieve details about an object
- Sync - to send a current snap shot of data

Enterprise Business Messages (EBM) are the operation-specific snippets of data that correlate to an Application Business Connector. These are subsets of the EBO and every componenet in an EBM should be included in the primary EBO. An example of an EBM would be the data that is required to create a person. This might include some of the same information that terminating a person would entail, but some elements would be different as well.

The glue that pulls everything together is the Enterprise Business Service (EBS) . EBS are the sexy part of the interface process where the two systems meet, and the typical activities we think of as interfacing happen. The more rigorous data validation and transformation happens here, as do things like Human Workflow, email notifications, and calls to any additional services that are required for the interface to occur.

Conclusion

While there is an initial investment in both the architecture setup for SOA-based interfaces, as well as learning to use the PeopleSoft Integration Broker and SOA Suite tools, this investment can be made with confidence. Oracle's stated direction for Fusion Applications is based on SOA and these technologies are non-proprietary and based on years of multi-vendor and public sector development.

[4] Taken from "Oracle® Application Integration Architecture Foundation 2.0: Concepts and Technologies Guide", Release 2.0, Part No. E10918-01.

Using PeopleTools for Seamless Integration with Oracle's SOA Suite and BPEL
April 2008

Authors: Dan Iverson, Solbourne
Chad Coffman, Solbourne

Solbourne
4001 Discovery Drive, Suite 210
Boulder, CO 80303
U.S.A.

Inquiries:
Phone: +1-303-417-2800
Fax: +1-303-417-2820
www.solbourne.com

This document is provided for informational purposes only and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by and means, electronic or mechanical, for any purpose, without our prior written permission. The Solbourne name, logo, related marks and all images on this document are owned by Solbourne and are registered and/or used in the U.S. All trademarks, service marks and trade names not owned by Solbourne referenced in this document are the property of their respective owners.