

University of New Orleans

ScholarWorks@UNO

University of New Orleans Theses and
Dissertations

Dissertations and Theses

Summer 8-2-2012

Integrating Oracle PeopleSoft Campus Solution to External Applications

Ishwor Aryal

University of New Orleans, iaryal@uno.edu

Follow this and additional works at: <https://scholarworks.uno.edu/td>



Part of the [Computer Sciences Commons](#)

Recommended Citation

Aryal, Ishwor, "Integrating Oracle PeopleSoft Campus Solution to External Applications" (2012). *University of New Orleans Theses and Dissertations*. 1508.

<https://scholarworks.uno.edu/td/1508>

This Thesis-Restricted is protected by copyright and/or related rights. It has been brought to you by ScholarWorks@UNO with permission from the rights-holder(s). You are free to use this Thesis-Restricted in any way that is permitted by the copyright and related rights legislation that applies to your use. For other uses you need to obtain permission from the rights-holder(s) directly, unless additional rights are indicated by a Creative Commons license in the record and/or on the work itself.

This Thesis-Restricted has been accepted for inclusion in University of New Orleans Theses and Dissertations by an authorized administrator of ScholarWorks@UNO. For more information, please contact scholarworks@uno.edu.

Integrating Oracle PeopleSoft Campus Solution to External Applications

A Thesis

Submitted to Graduate Faculty of the
University of New Orleans
in partial fulfillment of the
requirements for the degree of

Master of Science
In
Computer Science

By

Ishwor Aryal
B.S., Colorado Technical University, Denver, CO 2006

August, 2012

ACKNOWLEDGEMENT

I would like to thank my Professor Dr. Shengru Tu for giving me opportunity to work on this project and his excellent guidance and feedback without which this thesis would not have been possible. I would like to extend thanks to Dr. Golden G. Richard, III and Dr. N Adlai A Depano for being part of my thesis defense committee.

I would also like to thank University Computing and Communication team at UNO for providing infrastructure to work on and all the support they have provided.

Table of Contents

List of Figures	v
Abstract	vii
Chapter 1 Introduction	1
Chapter 2 Background	3
2.1 Oracle PeopleSoft Campus Solution	3
2.1.1 Application Designer	3
2.1.2 Security in PeopleSoft	3
2.1.3 Application Messaging and Web Service	5
2.2 Web Services Description Language (WSDL)	6
2.3 SOAPUI	7
2.4 SharePoint and Document Library	7
2.5 Microsoft InfoPath	7
2.6 Visual Studio	8
Chapter 3 Integration Design	9
3.1 Three Types of Integration	9
3.1.1 Application Messaging	9
3.1.2. Staged or Flat File	10
3.1.3. Web Service	11
3.2 Integration Broker	12
3.2.1 Integration Gateway	13
3.3 Inbound and Outbound Request Flow	14
3.4 Web Service Design based on Component Interface	16
3.5 Summary	18
Chapter 4 Implementation	20
4.1 Component Interface Construction	20
4.2 Configuration for Building Web Service	23
4.3 Building and Providing Web Services	24
4.3.1 Enabling Required Service Operation	24

4.3.1 Providing Web Service.....	26
Chapter 5 Case Study.....	30
5.1 Case I – Name Change Request.....	30
5.1.1 Problem Description.....	30
5.1.2 Configuration.....	31
5.1.3 Building Component Interface	32
5.1.4 Creating and Exposing Web Service.....	33
5.1.5 Testing Web Service.....	33
5.1.6 Implementation on InfoPath Form	35
5.1.8 Applying SharePoint Workflow Foundation.....	36
5.2 Case II – Retire or Rehire Employee in PeopleSoft Human Resource Component	40
5.2.1 Problem Description.....	40
5.2.2 Configuration.....	40
5.2.4 Building and Exposing Web Service.....	42
5.2.5 Testing Web Service.....	42
5.2.6 Implementation.....	45
5.2.7 Alternative Proposed Solution.....	45
Chapter 6 Conclusion and Future Work	47
References.....	48
VITA.....	50

List of Figures

Fig: 1 Peoplesoft Security Foundation.....	4
Fig: 2 Peoplesoft Security – User, Roles And Permission Lists.....	5
Fig: 3 High Level Messaging Architecture.....	10
Fig: 4 Message Flow In Two Applications.....	12
Fig: 5 Integration Gateway Architecture	14
Fig: 6 Flow Of Inbound Request	15
Fig: 7 Flow Of Outbound Request.....	15
Fig: 8 Ps Component Interface Architecture	17
Fig: 9 Application Designer: Project, Component And Component Interface View	21
Fig: 10 Ci Based Services Step 1	25
Fig: 11 Ci Based Services Step 2.....	25
Fig: 12 Web Service Wizard.....	26
Fig: 13 Web Service Wizard.....	27
Fig: 14 Web Service Wizard.....	27
Fig: 15 Web Service Wizard.....	28
Fig: 16 Web Service Wizard Results	28
Fig: 17 Test Case Project Pane	31
Fig: 18 Project, Component And Component Interface Side By Side View.....	33
Fig: 19 Soapui Wsdl Import.....	34
Fig: 20 Testing Person Search Method.....	34
Fig: 21 Infopath Form.....	35
Fig: 22 After Submitted	36
Fig: 23 Workflow Foundation In Sharepoint For Form Submission.....	37
Fig: 24 Sharepoint Document Library For Form Submit	37
Fig: 25 Workflow Foundation In Sharepoint For Form Approval	38
Fig: 26 Sharepoint Document Library View After Form Approved	38
Fig: 27 Email Triggered After Form Is Approved.....	39
Fig: 28 Peoplesoft Data Change View After Form Is Approved.....	39
Fig: 29 Component And Component Interface For Job Data.....	41
Fig: 30 Soap Wsdl Import.....	42
Fig: 31 Get() Method Of Web Service And Result In Soapui	43
Fig: 32 Status Of Employee: Rehired And Active Since 02/26/2012	43
Fig: 33 Testing Of Update Method In Soapui	44

Fig: 34 Peoplesoft User Interface Of Employment Data After Update Method Is Executed..... 44

Abstract

An integration solution must sustain multiple PeopleSoft upgrades, which is necessary to preserve investment in system integrations. Since the underlying structures and connection technologies of PeopleSoft have been and can be migrated from version to version in order to enhance features and performance, it is critical for any external component of integration to be built based on publicly visible interfaces of the PeopleSoft component. We have developed a standard-based solution to integrate “PeopleSoft Campus Solution” into “Microsoft SharePoint” using Web services generated by PeopleSoft’s Pure Internet Architecture. We have illustrated such kind of integration in two examples that emulate some of the imminent problems in the University’s current information systems between the PeopleSoft Campus and SharePoint Workflow. The methodology used in this is applicable to integrations of general COTS software systems into modern enterprise information systems.

PeopleSoft Campus Solution, COTS Software, Microsoft SharePoint, InfoPath, SOAPUI, Component Interface based Web Service, Integration Broker

Chapter 1 Introduction

Since the diversified business need of large companies and organizations always exceed any single commercial-off-the-shelf (COTS) software product, distributed systems consisting of multiple COTS products mixed with customized software are the common structures of enterprise information systems. Mounting experience has shown that the effective use of COTS software products has long been challenging since its inception decades ago (Dean, 2002). From the COTS-based Software System initiative advocated by Carnegie Mellon Software Engineering Institute (SEI) in 2001 to the theory and practice of “System of Systems”, significant efforts have been invested in research on COTS software integration (Fisher, 2007).

In principle, the Web service technology should be inherently suitable to system integration. In reality, proprietary techniques in COTS products often complicate integration. For example, Oracle PeopleSoft Campus Solution is one of the fastest growing products for Higher Education. In over two decades, PeopleSoft was difficult to connect to third party applications (Lynn, 2001). After Oracle acquired the product in 2004, a number of enhancements have been done to make the product able to communicate with external applications with various techniques (Lynn, 2001). However, it is still a challenge by experiments for developers and end users to make integrations. In the recent several versions of the PeopleSoft products, PeopleSoft has been promoting the Web service technology for interacting with external applications.

In this project, we have developed a systematic, standard-based solution to integrate “PeopleSoft Campus” into “Microsoft SharePoint Workflow Foundation” using Web services generated by PeopleSoft’s Pure Internet Architecture (PIA). The primary objectives of this project are to identify the available techniques for integration with Web services and minimize customization in integration. A critical criterion of success is to achieve version neutral. That is, an integration solution must sustain multiple PeopleSoft upgrades. This is necessary in order to preserve investment in system integrations. PeopleSoft upgrades their products periodically to enhance features and performance. PeopleSoft is obliged to facilitate migration of customers’ data. However, PeopleSoft does not support the transformation of clients’ customization code. Thus, massive code customization in PeopleSoft deployment will hinder future PeopleSoft upgrades (Tu, 2002). The underlying architecture and connection technologies of PeopleSoft have been migrated from the traditional client/server paradigm to a multi-tier architecture then to a service-oriented architecture. For interaction with a PeopleSoft component, it is critical for any external component to be built based on the publicly visible interfaces of

the PeopleSoft component. Such kind of visible interfaces could be available in PeopleSoft API or could be created using PeopleSoft PIA. If a change happens in the PeopleSoft API or the subject PeopleSoft component, PeopleSoft will be obligated to document the change which can assure manageability of the related integration parts.

The methodology that brought up the above solution to PeopleSoft-SharePoint integration is applicable to integrations of general COTS software systems into modern enterprise information systems. At the same time, the experiments carried out in this project have been targeted at a number of imminent problems in the University's current information systems between the PeopleSoft Campus and SharePoint Work Flow. An immediate goal of this project is to provide practical, affordable, manageable solutions to University's information systems. The ultimate goal is to advocate the integration approach based on Web-services in the System of Systems research (Fisher, 2007; Luzeaux, 2011).

The remainder of this thesis is assumes the following organization. Chapter 2 provides the background of the project with some technologies and specific components within PeopleSoft Campus Solution. Chapter 3 provides integration techniques and design. Chapter 4 provides implementation of chosen technique in chapter 3. Chapter 5 provides detail implementation case study. Two tests were done during the project: one with PeopleSoft Delivered Component for Human Resource Job related component and the other test case with custom built component Interface. The final chapter provides summary of achievements from the project along with conclusion.

Chapter 2 Background

Some of the technologies and products used for this thesis project are described in this chapter

2.1 Oracle PeopleSoft Campus Solution

Oracle PeopleSoft Campus Solution (formally known as PeopleSoft Campus Solution) is a comprehensive solution for student administration for Higher Education institutions. PeopleSoft was founded in 1987 with human Resource product as their first launched product and extended their service to variety of areas such as financial Institutions, healthcare Managements, education and public sectors organizations. The core components of this product are web server, application server, database server. The back end is oracle database and the front end is a browser. Within the application, core building blocks are fields, records, pages, menus, components, component interface, Application Engine (most of them are programmed in COBOL) (PeopleBook, 2012).

2.1.1 Application Designer

This is the core development tool to develop and customize PeopleSoft Applications for the PeopleSoft Pure Internet Architecture, maintain data, and perform updates and upgrades PeopleSoft Applications. It enables developers and system analyst to build a variety of definitions including Fields, Records (tables), Pages, Components, Menus, Component Interface, Testing, Program Debugging, Data Consistency Checking, Program Validation and write programs and modules in PeopleCode and SQL. This tool is used as developmental and testing in this project (PeopleBook, 2012).

2.1.2 Security in PeopleSoft

Security is especially critical for core business applications, such as PeopleSoft applications. Typically, what is needed is a need to restrict the usage, viewing and customization of the data and applications. PeopleSoft provides security features, including components and People Tools applications, to ensure that the sensitive application data do not fall into the wrong hands. As the PeopleSoft Internet Architecture (PIA) is implemented, a robust and scalable means is needed by which the users can be grant authorization efficiently.

Security can be applied to all users, including employees, managers, customers, contractors, and suppliers. Users are grouped according to roles given to them with different degrees of access. For instance, there might be an Employee role, a Manager role, and an Administrator role. Users who belong

to a particular role require a specific set of permissions, or authorizations, within the system, so that they can complete their daily tasks.

The objects and definitions in the PeopleSoft development environment must also be secured from viewing. Restriction can be implemented to block the end users from accessing particular pages and components, also to restrict the definitions that the site's developers can access using PeopleSoft Application Designer. A definition refers to any of the definitions that are created within PeopleSoft Application Designer, such as records, pages, or components. Each object definition may have individual security needs (PeopleBook, 2012).

Accessing a PeopleSoft application requires first passing through several layers of network, Operating System, and Database security. These capabilities are defined by the technical environment and need to be configured outside of PeopleSoft. The following figure demonstrate the same.

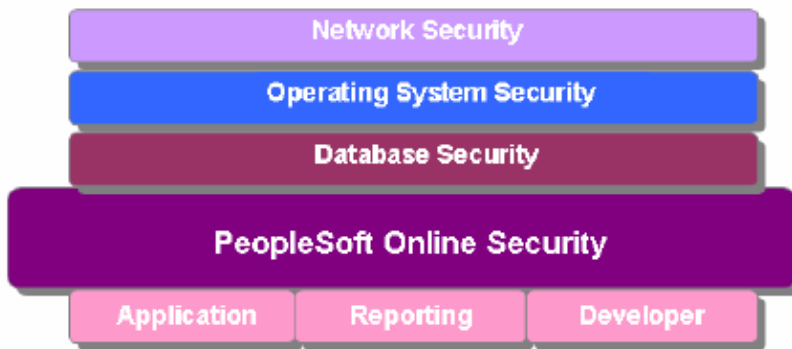


Fig: 1 PeopleSoft Security Foundation

The picture below exemplifies the relationship between Users, Roles and Permission Lists. Permission lists are assigned to roles, which are then assigned to user profiles. A role may contain numerous permissions and a user profile may have numerous roles assigned to it. Because permission lists are applied to users through roles, a user inherits all the permissions assigned to each role to which the user belongs. The user's access is determined by the combination of all of the roles.

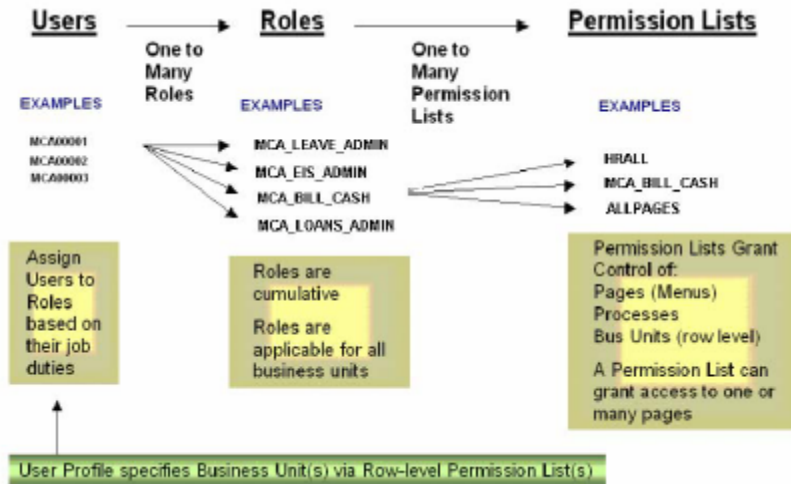


Fig: 2 PeopleSoft Security – User, Roles and Permission Lists

2.1.3 Application Messaging and Web Service

Application Messaging, based on the “publish-and-subscribe” model, allows PeopleSoft applications to integrate with each other and with third party applications. This model provides integration that is close to real-time, which means that the publisher need not be connected to the subscriber when publishing the data. On one end, a message is created and published and on the other end, the message is delivered to any number of subscribers.

Application messages are the fundamental building blocks comprising the application messaging system. Messages are self-describing entities formatted in XML. Each message contains data to be distributed among systems at runtime. To enable the creation and delivery of application messages, following object types in Application Designer are defined:

Message definitions: Stores the information about how a single application message is constructed. Each message definition has a multi-level structure, similar to components, that defines the data to be inserted into the application message at runtime.

Message channels: Channels correspond to groups of message definitions. They help order messages, enhance scalability, and provide a simple way to define processing characteristics of many similar messages as a single group. Channels include message routings, which define the mappings between message nodes on the messaging network.

Message Nodes: The physical systems (application servers or databases) connected to the messaging network.

PeopleSoft supports providing WSDL documents to the PeopleSoft WSDL and Universal Description, Discovery, and Integration (UDDI) repositories. The system enables consuming WSDL documents from other PeopleSoft and third-party systems, and automatically creates integration metadata based on the consumed WSDL documents for processing integrations. Other than basic building blocks, PeopleSoft WSDL document includes: WS-Security elements, UsernameToken and SAMLToken, PartnerLinkType elements (for BPEL Application) (PeopleTool 8.51, 2011).

A SOAP Message is an ordinary XML document that consists of three sections:

A SOAP Envelope – The top element of the XML document representing the message and defines the content of the message. It defines the framework of what is in a message, how to process it, who should deal with it and whether it is optional or mandatory.

A SOAP Header – This section is optional. It contains header information and attributes that can be set to encode and further identify the type of processing and additional features of the message.

A SOAP Body – This section contains the call and response information intended for the recipient of the message. This is the message “payload”. SOAP is not a requirement for exposing a web service, but should be used when appropriate when implementing a web service. Many web services today expose functionality over XML/HTTP without using SOAP.

2.2 Web Services Description Language (WSDL)

The shape and contents of the SOAP XML will not be the same for each transaction. One transaction might require one input parameter, while another might require many. The structure and contents of the request XML must be communicated to other parties who want to invoke a given web service. Likewise, the structure and contents of the response XML that is sent back with the results of the transaction need to be communicated as well. WSDL is an XML-based description of how to connect to a web service. It references a schema which describes the inputs and outputs of a web service and the URL to post requests to in order to invoke the web service (Benz, 2003).

The default URL format is as follows:

http://localhost/PSIGW.war/PeopleSoftServiceListeningConnector/PT_WORKLIST.1.wsdl

2.3 SOAPUI

SOAPUI is a free and open source cross-platform Functional Testing solution. With an easy-to-use graphical interface, and enterprise-class features, SOAPUI allows developer/testers to easily and rapidly create and execute automated functional, regression, compliance, and load tests. Most of the testing of the WSDL file generated by PeopleSoft Integration Broker is tested with this tool (SmartBear Software, 2011).

2.4 SharePoint and Document Library

SharePoint is a multipurpose tool to cater to web requirements common for most industries. Unlike a typical web application or content management system, SharePoint platform is designed to manage web server configuration itself. This allows for the bulk management, scaling, and provisioning of servers often required by large organizations or cloud hosting providers. A Library is a list where each item in the list refers to a file that is stored in SharePoint that contain files. Files created with library can be view in Browser as well and InfoPath application. Microsoft SharePoint comes with some pre-defined list and library definitions. These include: Announcement Lists, Blogs, Contacts, Discussion Boards, Document Libraries, External Content Lists, Pages, Surveys, and Tasks. With Microsoft Office SharePoint, workflows can be built that add application logic to the site/application/document without having to write custom code. Using the Workflow Designer, rules can be created that associate conditions and actions with items in Microsoft SharePoint lists and libraries, so that changes to items in lists or libraries trigger actions in the workflow. In this project, document library and workflow features are used as a Web service consumer and one of the data repositories respectively (Bustamante, May 2008).

2.5 Microsoft InfoPath

Microsoft Office InfoPath is a software application for designing, distributing, filling and submitting electronic forms containing structured data. It supports SOAP web service, REST web service, SharePoint Library or List, Database connection and XML documents. In this project, SOAP Web service is consumed by InfoPath form and utilizes workflow in SharePoint to route and save document in SharePoint Library as well as consume PeopleSoft exposed Web service method to maneuver data in PeopleSoft database (Bustamante, May 2008).

2.6 Visual Studio

It is an integrated development tool to develop applications in wide variety of programming languages and techniques. In this project, this tool is used to test some basic functionality of PeopleSoft Web service Methods exposed to external Application (Sempf, 2004).

Chapter 3 Integration Design

There are two types of integrations available in PeopleSoft; Real Time Integration and Non-real time Integration. Most of the integration done in PeopleSoft - to - PeopleSoft is done in real time. It is called messaging which is an xml based data exchange. All other integration is done in non-real time where data are dumped in a table or a flat file from one system and later picked up by other. This involves lots of resources writing the data dump program and processing program. In recent years, Web service (both WSDL and REST) was introduced in PeopleSoft to communicate with external application. Because of technical complexity, product limitations, and lack of proper documentation, this technique has been mainly used by Software consulting companies but has rarely been successfully used by PeopleSoft user community. In this chapter, PeopleSoft's integration types, its limitations and challenges as well as work around are discussed in detail. In this chapter we further study Component-Interface based Web Service.

3.1 Three Types of Integration

There are three major methods of integration available in PeopleSoft Campus Solution:

- Application Messaging
- Staged or Flat File
- Web Service

3.1.1 Application Messaging

“Application Messaging” is an XML based solution. The standard procedure is to build nodes, build messages and then finally create channel for those messages. Application messaging guarantees that messages are delivered in the order published and that they are single-threaded on the subscriber. Therefore, it is the publisher's responsibility to publish messages in the proper order. Application messaging supports the integration of PeopleSoft applications and third party systems by publishing business events as XML messages. To publish data, third party applications can post XML messages directly to PeopleSoft's Application Messaging Gateway servlet. To subscribe to data, third party applications can accept and process XML messages posted by PeopleSoft by adding a custom Java subscription handler to the Application Messaging Gateway servlet. To determine whether the publish was successful, standard HTTP Post return codes and the PeopleSoft Reply Document return code can be checked. If the subscribing PeopleSoft system is not available at the time the third party system attempts

the publish, the message will not be delivered. The third party system must queue the messages and “retry” until the subscribing system is back online (PeopleTool 8.51, 2011).

This solution works but has lots of overhead to achieve the goal. This process needs lots of XML parsing and generation of PeopleSoft proprietary schemas. This method is useful when organization share data between different information systems internally, and with partners’ external systems. Application messaging provides benefits including synchronization of data between systems and System-to-system workflow. There are scenarios in which application messaging is advantageous such as Application-to-application integration and Cross-release integration

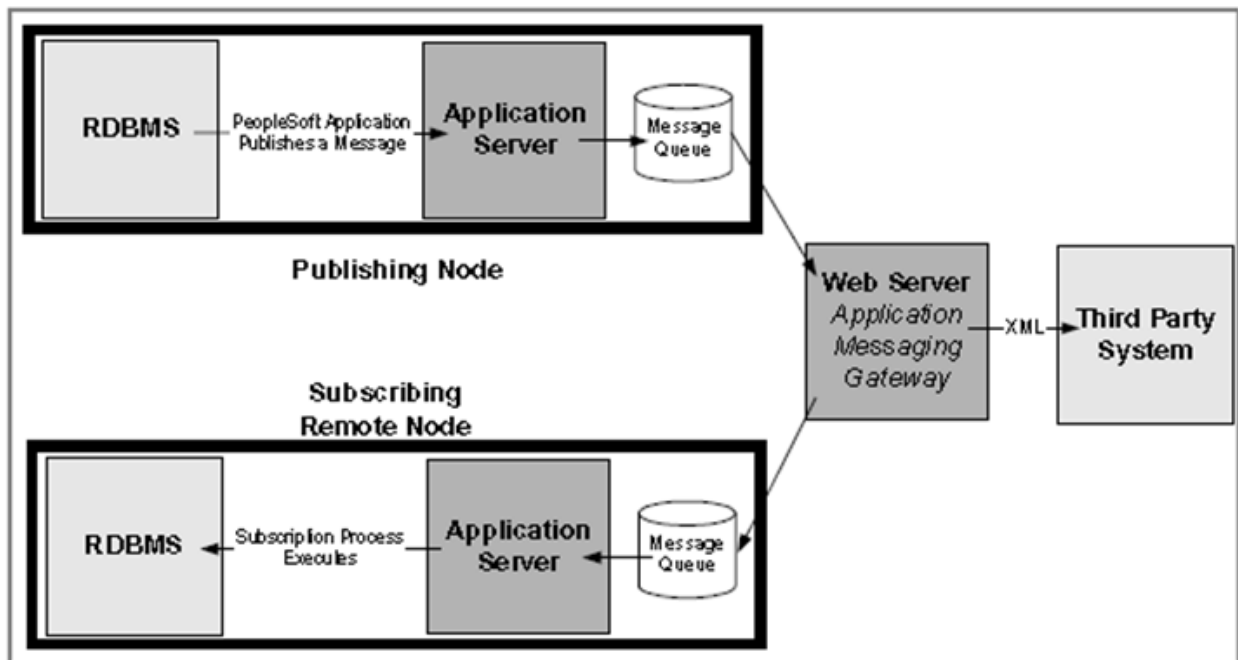


Fig: 3 High Level Messaging Architecture

3.1.2. Staged or Flat File

This is a non-standard way of integration. This is not real time integration. This process skips existing implementation of business rules but depends on PeopleSoft internal data schema. The data elements that are needed for the integration is generated with SQR and SQC (PeopleCode) and/or SQL and stored in flat file or database. Later same database or file is used to update data in another system using PeopleCode or other application programming languages. The critical problem with this technique is the dependency to PeopleSoft’s internal data schema. From version to version, the PeopleSoft products can reasonably change their internal schema. When this kind of change happens, the investment of integration suffers the danger of dysfunction. Sometimes this process is useful if the third party vendor

only provide flat file or CSV file or other sorts of database structures. This technique has been superseded by a later technique called file parser.

3.1.3. Web Service

The PeopleSoft defines that a Web service is simply an application component that is accessed programmatically over the internet using XML over HTTP. Any discrete component of application functionality can be exposed as a Web service. Examples include Employee names, Job data, and customer profile. Any of these application components can be published and accessed over the internet as web services. Web services consist primarily of the following standard technologies:

- XML
- SOAP
- WSDL

XML is the language in which all the data structures, SOAP and WSDL are written. They are the most basic requirements for a Web service. Simple Object Access Protocol (SOAP) specifies the format and schema of messages. WSDL specifies the operations of Web services. SOAP and WSDL make cataloging, discovering and implementing Web service easier.

Web services are loosely coupled as they have well defined, published interfaces in WSDL and can be easily accessed from remote systems over the internet by (HTTP). They require coordination based on standards. The underlying technology behind the Web service can be changed and replaced without impacting the standard-binding components for integration. This loose coupling nature of Web services decouples the PeopleSoft internal data schema and external applications, which make possible for software integration to sustain PeopleSoft upgrade progress, to preserve the investments of integration, and making it easier to integrate applications (Benz, 2003).

All PeopleSoft components can invoke or can be invoked as Web services using the Integration Broker. For example, following diagram summarizes the message flow of PeopleSoft Supply Chain Management (SCM) submitting a Customer Profile Web service request in PeopleSoft Customer Relationship Management (CRM).

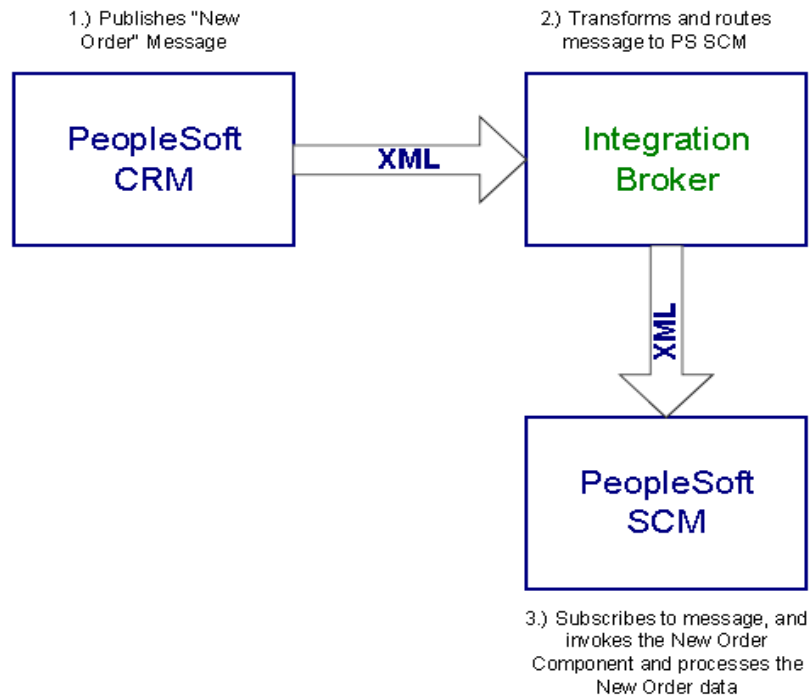


Fig: 4 Message Flow in two Applications

In figure 4, PeopleSoft Customer Relationship Management submits a request to create a “new order” using Web service, the service operation defined in Integration Broker is executed. It transforms and reroutes the incoming messages to PeopleSoft Supply Chain Management. Internal application in Supply Chain Management then subscribes the messages; invoke the “new order” component of PeopleSoft Customer Relationship Management. Outbound would be similar except Supply Chain Management System executes the method and return response to Integration Broker that sends response back to Customer Relationship Management.

Most discussions around Web services are based on synchronous style integration. But sometimes the calling program does not want to incur the overhead of synchronous calls to remote systems Using PeopleSoft Application Messaging; web services can be invoked asynchronously.

3.2 Integration Broker

PeopleSoft Integration Broker facilitates integrations with other PeopleSoft and third-party systems. It features a services-oriented architecture that enables developers to expose PeopleSoft business logic to calling components. The PeopleSoft Integration Broker is a framework that supports synchronous and asynchronous messaging in a variety of communication protocols. It manages message structures, message contents, and transport disparities. The Integration Broker also has native SOAP support for

sending and receiving messages. Using the Integration Broker technologies, PeopleSoft applications can be both Web service clients and a Web service server. Web services in PeopleTools are built from the bottom-up approach starting with the message that needs to be passed to the caller of the Web service. The same messages are used while defining new service operations which are grouped into Services. The service operations are then provided with PeopleCode Handlers to take the inbound request messages or populate outbound response messages (PeopleTool 8.49, 2011).

The two major components of PeopleSoft Integration Broker are the integration gateway and the integration engine. The integration gateway is a platform that manages the receipt and delivery of messages passed among systems through PeopleSoft Integration Broker. The integration engine is an application server process that routes messages to and from PeopleSoft applications as well as transforms the structure of messages and translates data according to specifications defined.

3.2.1 Integration Gateway

The structure of the Integration Gateway is shown in Figure 3.2. Its three components are explained below.

Connectors: Listening and Target connectors to receive and deliver messages. Listening connectors receive requests from integration participants, send them to the gateway manager, and deliver responses back to the integration participants. The target connectors generate requests, send them to integration participants, wait for responses from participants, and deliver the responses back to the gateway manager. The integration gateway invokes target connectors dynamically through the gateway manager. The target connectors adhere to a standard structure by implementing the target connector interface provided by the integration gateway. By implementing this interface, target connectors can take advantage of all gateway manager services.

Gateway Manager: The gateway manager processes all messages flowing through integration gateway and maintains links to all integration components including connectors and gateway service.

Gateway Services: These are the services that gateway managers for messages parsing. It uses two message objects: IBRequest and IBResponse to enter and exit Integration Broker. It also uses connector management services for message routing functions that has varying level of complexity. Web service security, error and service logging, error handling, and message validation are major functions of Gateway service that are ultimately used by gateway managers

The primary function of Integration Gateway is to manage the receipt and delivery of messages. Following are basic features of Integration Gateway:

- Listening and target connectors that transport messages between integration participants and integration engine
- Logging information regarding message receipt delivery and errors
- Checking format compatibility of the message

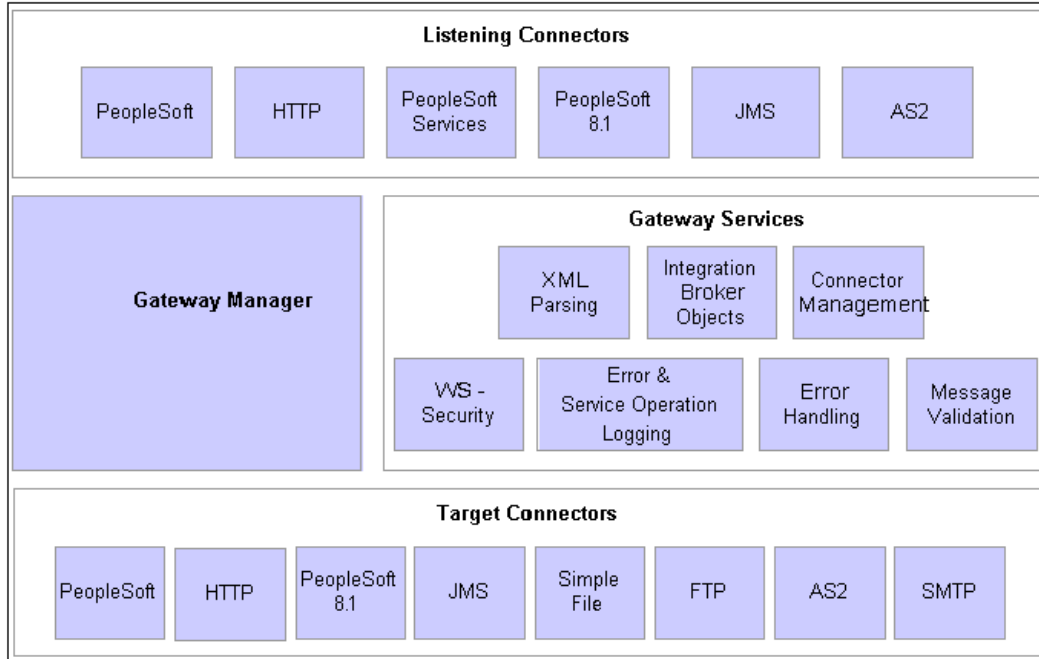


Fig: 5 Integration Gateway Architecture

In integration gateway, listening connectors receive messages and deliver to the gateway manager which determines which target connector to use to deliver the message to their intended recipient. The target connector then delivers the messages to the intended recipient.

3.3 Inbound and Outbound Request Flow

The purpose of the components explained earlier in this chapter is to support interactions in an integrated system, which is realized by inbound and outbound messaging. Figure 5 and 7 demonstrate the inbound and outbound requests within PeopleSoft Integration Broker. A request is sent into the gateway by an external application. Then the Integration Gateway passes it on to the application server. The

application server processes the request, and returns a response which is sent back to the external system by the Integration Gateway.

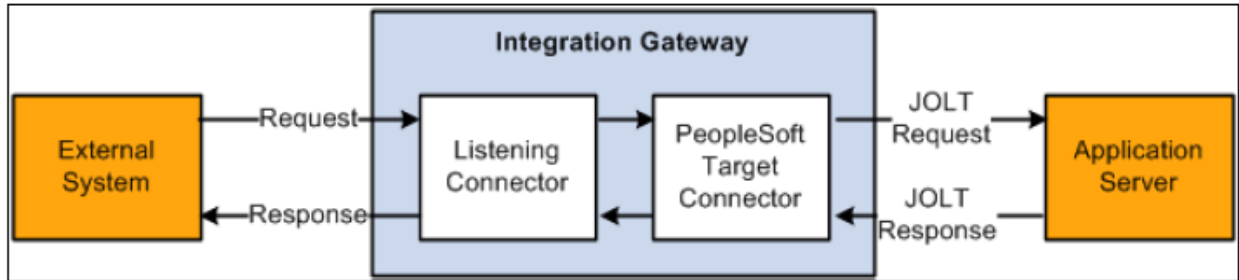


Fig: 6 Flow of Inbound Request

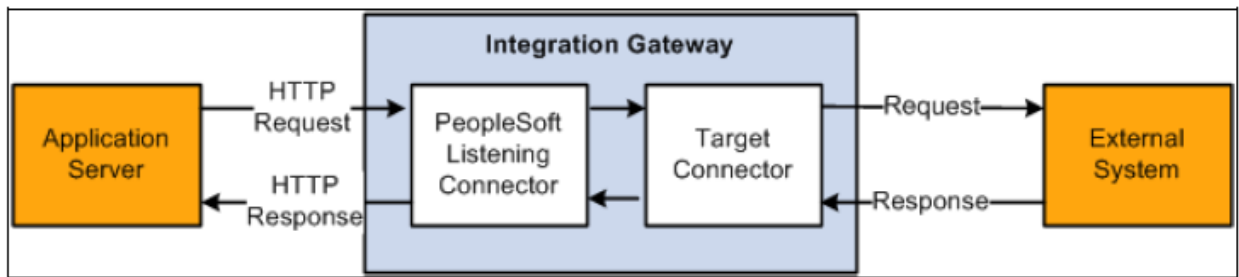


Fig: 7 Flow of Outbound Request

This process consists of the following steps:

Step 1: An external application sends a request to Integration Broker

Step 2: The request is received by the listening connector which writes the request to the gateway log file exactly the same as it was received on wire; normalizes the service operation for the use by application server, and populates an internal request class from the received request.

Step 3: The request is then processed by the PeopleSoft target connector which serializes the request string and sends to application server via a Java Online Transaction Connection (JOLT). All communications between the gateway and the application server is done via Multipurpose Internet Mail Extensions (MIME) messages. This MIME message is written in gateway log before sending to application server.

Step 4: Once the application server receives the MIME request, it parses the message into a request object and pre-processes it. Pre-processing is a multistep process to authenticate service operations, determining the direction of the operation (inbound/outbound), identifying connectors and its properties, and

extracting the information about the node and determining runtime handlers. Once these steps are done, further processing carried out based on the available data by PeopleCode behind the scene.

For the outbound request, once the output is ready, the application server converts message to the MIME standard format. The request must be sent to the PeopleSoft listening connector on the gateway using the value of the Gateway URL defined for the given gateway. If this URL is not valid or does not point to the PeopleSoft listening connector, the application server will be unable to send the request.

Step 5: Once the Application Server processes the request, regardless of the result, it sends response back to the gateway. If the connection between gateway and application server is synchronous, it will get some response only after the completion of the operation whereas if the connection is asynchronous, then integration gateway will receive response as soon as the delivery is confirmed. The response will be in MIME format.

Step 6: Once the response is received by PeopleSoft Target Connector, it is written in gateway log, parse the MIME back to gateway request object and return to listening connector.

Step 7: The final step is to pass response to external application by Listening Connector according to the protocol used.

In PeopleSoft Campus Solution, Web service can be built through 2 methods: Application Classes and Component Interface. Each has advantages and disadvantages however; this project focuses on building web service through Component Interface. (PopleBook, 2011)

3.4 Web Service Design based on Component Interface

A component interface is to enable exposure of a PeopleSoft component for synchronous access from another application (PeopleCode, Java, C/C++, COM, or XML). A Component interfaces can be viewed as "black boxes" that encapsulate PeopleSoft data and business processes, and hide the details of the structure and implementation of the underlying page and data. A component interface maps to one, and only one, PeopleSoft component. A component interface is created in the PeopleSoft Application Designer by selecting the PeopleSoft component. Record fields on the PeopleSoft component are mapped to the keys and properties of the Component Interface. Figure 8 shows the relationship between the basic elements of the component interface architecture.

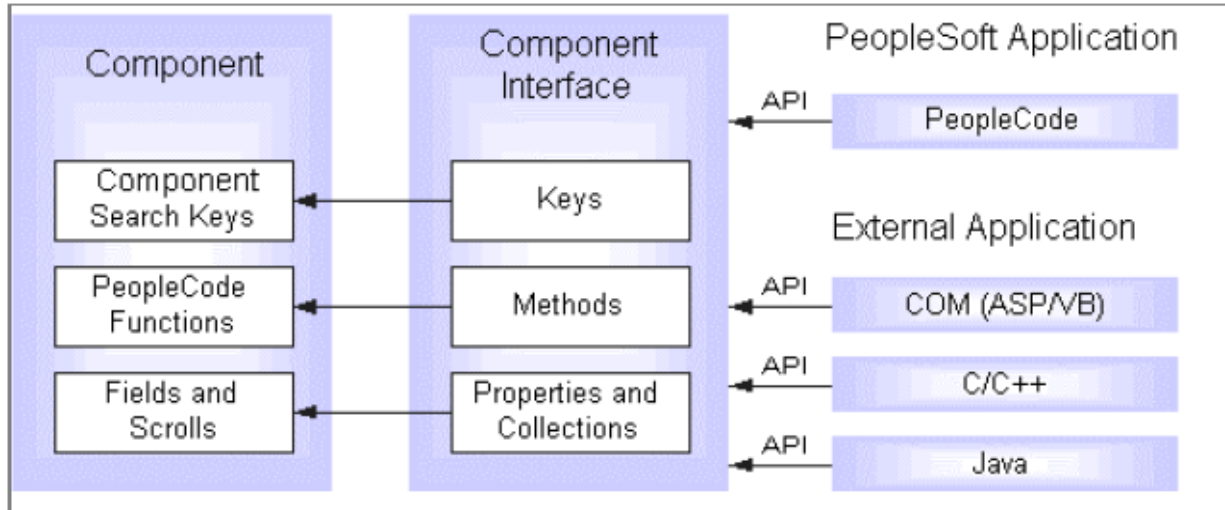


Fig: 8 PS Component Interface Architecture

Every component interface has the following main attributes:

- Component Interface Name
- Keys (get keys, create keys, and find keys)
- Properties and collections (fields and records)
- Methods.

In most cases, component interfaces behave the same as their associated components. This means that PeopleCode events typically trigger in the same order as the component. However, several runtime considerations cause exceptions to this behavior relate both to PeopleCode processing and search dialog box processing (PeopleTool 8.51, 2012).

This project highlights the web service based on Component Interface of PeopleSoft. Once the component is in place in PeopleSoft, a component interface is built in application designer. Web Service is then built based on the Component Interface. Every time the Interface is updated, Component Interface needs to be modified including the People Code behind scene if there are any modifications done in the delivered component. The final step is to recreate the web service and expose it. Component Interfaces are PeopleSoft's way of exposing the business logic developed into Components for consumption by other areas of the system. Component interfaces are part of PeopleSoft's Integration Broker technology and an attempt to introduce SOA into the product. They tend to work quite well but can be slow for large amounts of data processing. This solution is best suited if data needs to be insert/update/delete through PeopleCode, and PeopleCode requires replicating a lot of existing business logic that already exists in a component then a component interface is the best approach as all the business logics are already in place.

PeopleSoft can automatically generate a component interface for developers but have to note that it will not be completely correct. Some tweaking needs to be done a to meet requirements. Before going any further, validating component interface is the key which is again a PS delivered functionality/tool. There are five standard methods come with component interface depending upon the type of interface and its functionality.

Cancel: Cancels the instance of the component interface object executing this method and rolls back changes. This closes the component interface and returns its *created* state. Returns True if component is successfully cancelled, otherwise returns False.

Create: This creates (adds) a new set of keys to the component - essentially the same as pressing add and entering the relevant keys through the component. The created keys are associated with the component interface object. At this point the CI is instantiated with the created data. Returns True if data is successfully created, otherwise returns False.

Find: Find allows for a partial key (wildcard) search for data in the underlying component.

Get: Gets the data from the underlying component interface matching on the get keys specified? The component interface is object is instantiated with the resulting data from the component. Returns True if data is successfully retrieved, otherwise returns False.

Save: Save changes made to the component data through the component interface object. The saved method triggers standard PeopleCode processing. All errors are logged to the PeopleSoft Messages Collection (PSMessages) and this collection may be used for troubleshooting errors.

3.5 Summary

Among couple ways to integrate PeopleSoft Campus Solutions to external applications, this project focuses on the method of integration using Web service. Web service can be built in couple ways but this project discusses creating Web service using components in PeopleSoft. Since a component is a collection of records, keys, and PeopleCode functions logically grouped together, it provides complete business logic to be used out of the box even with limited understanding of the underlying processes that takes place within PeopleSoft System when the component is used.

I have devised a solution to allow outside applications such as Microsoft InfoPath to update the PeopleSoft database through an existing PeopleSoft component as if a user keys in the data through a PeopleSoft page. With this method, every business rule is still enforced by the original PeopleSoft

component. Such integration can sustain PeopleSoft upgrades if the interface of the related component is not changed and any internal data schema change will not affect the integration. Every problem that is possibly caused by an upgrade can be examined by reviewing the interface of the related components only.

Chapter 4 Implementation

This chapter describes the implementation of the design presented in chapter 3. As discussed in chapter 3, the integration method used is based on Component Interface. This project has some pre-conditions regarding the preparation for the implementation. Following are some preconditions:

- **Components:** A set of pages and business logic for (the business purpose) is identified in the existing PeopleSoft System or created by the programmer
- **Component Interface:** PeopleSoft delivers few built-in Component Interfaces; if not delivered it needs to be created using Application Designer.
- **Access privilege for Components:** Proper PeopleSoft security needs to be provided in order to use component. For this project, the Administrator-level security is assigned.

4.1 Component Interface Construction

Once the component is built and ready to use, Component Interface is built off with that Component. A component interface maps to one, and only one, PeopleSoft component. The following figure shows the side by side view of Component and Component Interface. Left pane is project pane where all elements of the project such as fields, records, pages, menus, component name, and component Interface name along with project name are displayed. In the middle pane is Component Pane where all the properties and records related to the component is displayed. The right pane shows the Component Interface built off of the Component selected. Even though five methods are displayed on the Component Interface pane for the selected component, only some methods are displayed for all component Interfaces. For example, PeopleSoft does not allow delete method to be exposed for any components to external applications.

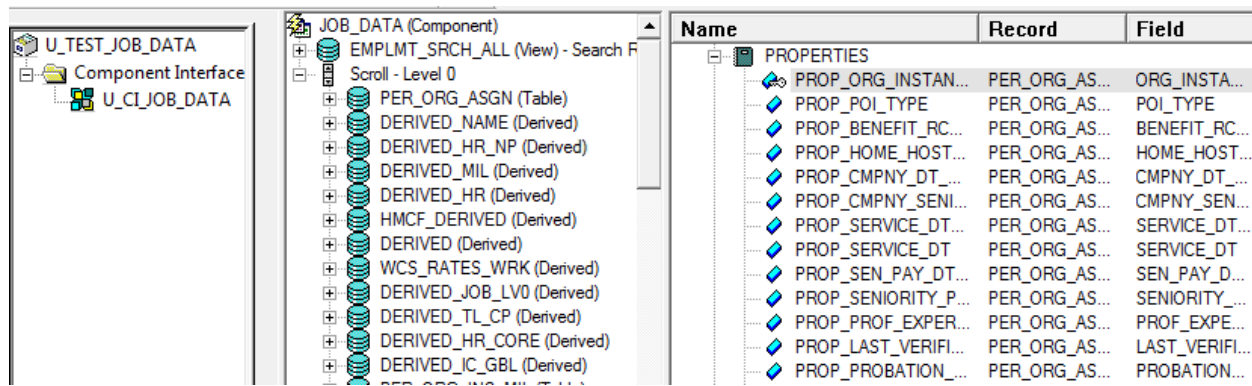


Fig: 9 Application Designer: Project, Component and Component Interface view

A component Interface view shows records and scrolls in the component via tree representation. A scroll in a component interface is referred to a data collection. A data collection is the generic APIObject that acts as a container of the collection and of other PeopleCode objects. Data collection objects are used to store the data in a scroll level so that it can be programmatically traversed.

Among several ways of building Component Interface, this project uses the PeopleSoft's delivered way to create Component Interface using Application Designer. PeopleSoft offers five attributes for each Component Interface namely:

Name: Unique name of the Component Interface specified while saving it. This name is later used by programs that call the Component Interface to access its properties and methods.

Key: Special properties containing values that retrieve an instance of list of instances for the Component Interface. It is created by PeopleSoft based on the search record definition for the underlying component.

Properties: Each property includes both component data and Component Interface settings. There are two kinds of properties. The Standard properties are assigned automatically when Component Interface is built; the User-defined properties are defined by developer mapped to record fields on the PeopleSoft component and are displayed in Application Designer. Properties can correspond to either record field or scroll (a collection of records).

Collections: Each collection is a special type of property that corresponds to a scroll. It contains fields and subordinate scrolls as defined in its underlying component. By default, a collection uses the name of the primary record for the underlying scroll.

Methods: Methods are functions that do specific tasks. The standard methods are those available for all Component Interface. The User-defined are the methods added by developer to add functionality to existing Component Interface.

The construction a Component Interface will require the following steps.

Step 1: Determining the Fields to Expose in Component Interface

Fields from a component are exposed in the component interface by dragging a record field or scroll from the component view into the component interface view shown in figure 3.1. Thorough understanding of the underlying component is required so that exposed fields are actually required in the external application. For example, if the component has a field called National_ID, before exposing it to the Component Interface, requirement of that field in external application should be verified. The idea is to expose only those fields that are necessary. The component view displays fields that are available in the component buffer at runtime. Buffer is the place in a component where all the fields are loaded at once for that component at runtime. For example, if a record containing 10 fields has only 3 fields included on a page or visible to the page, then the component view will list only those 3 fields however it loads all 10 fields in buffer (PeopleBook, 2012).

The first time a collection from the component view is dragged to the component interface view, the system uses the following rules to determine what properties to expose:

- Keys are exposed only at the highest-level scroll (data collection) in which they first appear. Typically, Get keys or Create keys are not exposed as properties, because these are set before a Get or Create operation and might be inadvertently changed.
- Make sure that all properties within the collection are not deleted; that would result in an empty collection. If such empty collections exist, they need to be removed; otherwise, they appear with X in the component interface view.
- If the page does not support Add mode, the level-zero record of the component should not be exposed, because it contains data that is not specific to the component interface that is being created.
- Fields that is not visible in the component view should not be exposed.

The component optimization code (execute within Application Engine) may attempt to eliminate unused fields from its buffer, which could result in an error if that field is accessed by the component interface.

Step 2: Validating Component Interface

Validation ensures that the structure of a component interface is still valid. Over time, the structure of a component interface can become invalid due to component structural changes and

modifications. For example, this can happen whenever a component deletes or adds a record or field. It can also happen if the keys on the component are added or removed. Properties and keys that no longer synchronize with their associated components are marked with an X icon. This is also checked in Application Designer tool.

Step 3: Component Interface Security

After creating a component interface, permission must be set for the new component interface and its all methods before it can be accessed. Security management for the component interface is provided through the PeopleSoft Internet Architecture (PIA) pages. As discussed in (chapter 2 to Component Interfaces) permission are set at the permission list level in PeopleSoft security manager.

Step 4: Testing Component Interface

PeopleSoft Application Designer delivers a simple test tool called Component Interface Tester. Once the component interface is ready, it can be tested for its intended functionality. Only methods that are exposed and have proper security setting can be tested using the tool. A result pane is displayed if the test is successful. Once the test is complete, Component Interface is ready to be used. For this project, web service is built off of this Component Interface and exposed so external application can consume its methods. Proceeding sections will provide details of Web service creation, security and process to expose to external application followed by consuming the service.

4.2 Configuration for Building Web Service

Web service is built with Integration Broker. Following are the pre-conditions for Web service creation and consumption. Integration Broker needs to be installed and configured for any external or internal application so that they can communicate with PeopleSoft. The following are some of the key elements that need to be setup before attempting to build Web service.

- Enable the publication and subscription processes in the application server so there is support for asynchronous services. These are the services required to provide queues for gateway to store service request. In this case, the communication is synchronous so this is not mandatory.
- Change the Default Local Node to some meaningful one. In this project PeopleSoft delivered generic node (PSFT_HR) is used.
- Configuration of the PeopleSoft Listening Connector is the first required step for this project. This is necessary for the Application Server to locate and communicate with an Integration Gateway.

- Configuration of the PeopleSoft Target Connector is also required so Integration Gateway knows where the Application Server is located.
- This is very important step to complete the service configuration. This is required for running web services. In this step, some default naming information for services and schemas are entered and also the PeopleSoft Service Listening Connector is identified.
- Other standard setup includes securing the integration environment by applying security at the web server, gateway, application server, nodes and service operation level; fine tuning integration system performance by employing failover, master/slave processing, load balancing

4.3 Building and Providing Web Services

Once the setup is tested and confirmed, Web service is built from PeopleSoft Integration Broker off of selected Component Interface. These are the steps of the process where service operations such as update, create, get and others can be selected to be exposed in Web service based on the component selected. Again, not all components expose all methods. Building and providing Web service is divided into two sections as following:

4.3.1 Enabling Required Service Operation

- Select the component Interface to be exposed as a Web service. This can be achieved by using search page in “CI-Based Services” menu in PeopleSoft Integration Broker component
- Review the Component Interface status to determine availability. For the first time it is used, the status will be “Does not exist” which means operations are not being used in any other services for that component Interface. The diagram below shows the status of each operation along with action type and methods. Selected (checked) methods will become service operation to be exposed in Web service that is being built.

CI-Based Services

Review Status

Status Find First 1 of 1 Last

CI Name: NJM_VNDR_ID **Description:**
Service: CI_NJM_VNDR_ID **Status:** Service does not exist and will be created.

Select	Action	Method	Service Operation	Status
<input checked="" type="checkbox"/>	Create Operation.	Get		Does not exist.
<input checked="" type="checkbox"/>	Create Operation.	Create		Does not exist.
<input checked="" type="checkbox"/>	Create Operation.	Find		Does not exist.
<input checked="" type="checkbox"/>	Create Operation.	Update		Does not exist.
<input checked="" type="checkbox"/>	Create Operation.	Updatedata		Does not exist.

[Select All](#) [Deselect All](#)

 [Return to Select CIs](#)

Fig: 10 CI Based Services Step 1

- Once “Display Selected Actions” is clicked, default (PeopleSoft System generated) operation names are degenerated along with message versions. Aliases and messages versions may be specified if required however the service operation name cannot be changed.
- Again, once the “Perform Selected Actions” is clicked, operation status is changed from “not created” to “Operation created”. This indicates that the Web service and Service Operations are now created and Web Service can be now published and WSDL can be generated. The following diagram illustrates the same.

CI-Based Services

Review Status

Status Find First 1 of 1 Last

CI Name: NJM_VNDR_ID **Description:**
Service: CI_NJM_VNDR_ID **Status:** Service saved.

[View Service Definition](#)

Select	Action	Method	Service Operation	Status
<input type="checkbox"/>	Create Operation.	Get	CI_NJM_VNDR_ID_G.V1	Operation created.
<input type="checkbox"/>	Create Operation.	Create	CI_NJM_VNDR_ID_C.V1	Operation created.
<input type="checkbox"/>	Create Operation.	Find	CI_NJM_VNDR_ID_F.V1	Operation created.
<input type="checkbox"/>	Create Operation.	Update	CI_NJM_VNDR_ID_UP.V1	Operation created.
<input type="checkbox"/>	Create Operation.	Updatedata	CI_NJM_VNDR_ID_UD.V1	Operation created.

[Select All](#) [Deselect All](#)

 [Return to Select CIs](#)

Fig: 11 CI Based Services Step 2

4.3.1 Providing Web Service

Creating Web service with PeopleSoft delivered wizard is the easiest and effective way. It is located under PeopleSoft Integration broker and has 4 steps process as following:

Step 1: Select Service to be provided (defined in section 4.3.1). Service name is system generated name similar to Component Interface name. It can be changed manually or Service alias can be added for external applications consuming that service.

Provide Web Service Wizard Step 1 of 4

1 2 3 4 Next >

Select Services

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

Search Criteria	
Service Name:	begins with <input type="text" value="CI_NJM_VNDR_ID"/>
Description:	begins with <input type="text"/>
Object Owner ID:	equals <input type="text"/>

Search Results		Find View All	First	1 of 1	Last
<input type="checkbox"/>	Service	Description			
<input checked="" type="checkbox"/>	CI_NJM_VNDR_ID	CI_NJM_VNDR_ID			

[Select All](#) [Clear All](#)

Fig: 12 Web Service Wizard

Step 2: Select Service Operation that is going to be exposed in Web service. This is the step of the process where decision for exposing methods takes place. For example in the following diagram, if update functionality needs to be blocked from the external application that is consuming the Web service, CI_NJM_VNDR_IP_UP.V1 should be unchecked.



< Previous

Next >

Select Service Operations

Select one or more operations for each service.

Service: CI_NJM_VNDR_ID		Description: CI_NJM_VNDR_ID			
Operations Find View All					
	Service Operation	Description	Operation Type	Request Message	Response Me
<input checked="" type="checkbox"/>	CI_NJM_VNDR_ID_C.V1	CI_NJM_VNDR_ID_C	Synchronous	M792311.V1	M267485.V1
<input checked="" type="checkbox"/>	CI_NJM_VNDR_ID_F.V1	CI_NJM_VNDR_ID_F	Synchronous	M492070.V1	M1095116.V1
<input checked="" type="checkbox"/>	CI_NJM_VNDR_ID_G.V1	CI_NJM_VNDR_ID_G	Synchronous	M282836.V1	M520300.V1
<input checked="" type="checkbox"/>	CI_NJM_VNDR_ID_UD.V1	CI_NJM_VNDR_ID_UD	Synchronous	M1021597.V1	M386507.V1
<input checked="" type="checkbox"/>	CI_NJM_VNDR_ID_UP.V1	CI_NJM_VNDR_ID_UP	Synchronous	M204495.V1	M763044.V1

[Select All](#) [Clear All](#)

Fig: 13 Web Service Wizard

Step 3: This is the final step of building Web service for the selected component Interface. In this step, WSDL is generated for the service. The WSDL file can be viewed with the system generated link.



< Previous

Next >

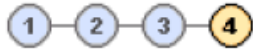
View WSDL

View the generated WSDL for each service.

Selected Services First ◀ 1 of 1 ▶ Last	
Service	Description
CI_NJM_VNDR_ID	CI_NJM_VNDR_ID View WSDL

Fig: 14 Web Service Wizard

Step 4: In this step, WSDL file publishing options are picked. It can be either published in WSDL Repository or published to UDDI servers. For this project, it is published in WSDL Repository.



< Previous

Finish

Specify Publishing Options

The WSDL for the selected services will be published to the PeopleSoft WSDL Repository. You can also publish the WSDLs to one or more UDDI Servers.

 Publish to UDDI WSDL Repository

Fig: 15 Web Service Wizard

Once these steps are completed PeopleSoft Integration Broker will provide a confirmation page with the generated WSDL URL that looks similar to the following diagram.

Provide Web Service Wizard

Confirm Results

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

WSDL Generation Log:

Service: CI_U_MY_TEST_CI has been exported.

Inserted WSDL: CI_U_MY_TEST_CI.1 in the repository

Generated WSDL URL:

https://cscsint.uno.edu/PSIGW/PeopleSoftServiceListeningConnector/PSFT_HR/CI_U_MY_TEST_CI.1.wsdl

Fig: 16 Web Service Wizard Results

Again, PeopleSoft security needs to be setup for each exposed operations as well as Web service. After applying proper security, WSDL URL can be distributed to any number of external sources to be consumed. Security to these PeopleSoft delivered operations is mandatory: GETWSDL and GETSCHEMA. These are delivered permissions for accessing WSDL file and schemas for newly created Web service.

One of the most important things learned during this project is that: just because WSDL is generated or Web service is generated does not mean it is valid. There are so many things could go wrong such as XML schema definition could not be validated and writing that definition could be difficult.

Thorough understanding of the product is required. Another challenge was to test the WSDL file that could not execute the INIT procedure in PeopleSoft. Some of the real time problems and challenges are discussed in Case Study chapter.

Chapter 5 Case Study

In this chapter, two experimental cases are presented. In the first case, everything is created from the scratch, which includes the PeopleSoft records, pages, components, component interfaces and the PeopleSoft Web service. In the second case, we use an existing component in the PeopleSoft Human Resource Management component called “Job Data”. This component includes all job related information for employee.

5.1 Case I – Name Change Request

5.1.1 Problem Description

PeopleSoft’s building blocks are fields, records (tables), components, pages, menus, and COBOL and PeopleCode associated with them for processing. In this test case, all those elements are created from the scratch in PeopleSoft using PeopleSoft Application Designer. This is a simple case where name change request is made by an authorizing department such as reception desk of Office of Admissions for existing student via an online form. Once a change request form is submitted, according to the predefined workflow, an email is triggered to a business unit. The document is stored in a secure location for further processing. An operator then reviews the form, approves or denies it and submits the form along with his/her decision. This submission triggers another workflow where the original initiator receives a notification of the action via email and the form is sent to a secure location for record logging. If the form is approved, the student’s name is updated in PeopleSoft Campus Solution by invoking update method using Web service. If the request is denied, the form is still stored in a secure location, but the student data is not updated in PeopleSoft Campus Solution. In this example, the external application consuming PeopleSoft Web service is Microsoft InfoPath form and the secure location is a Microsoft SharePoint Server.

When the InfoPath form is launched, the user is prompted to enter the student identification number. The form pulls the name from PeopleSoft using the Get() method exposed through a Web service and populates the form. When the change of name is approved by the operator, the InfoPath accesses the Update() method to update student’s name through the PeopleSoft Web service. In this section, we will discuss how the component, component interface and Web service are built, how security for component, components Interface, and web service are used.

5.1.2 Configuration

As discussed in Chapter 3 and Chapter 4, configuration is important. In this test case, configuration includes creation of required fields, record (table), page, menu, component, and security. The first step of the process is to identify the fields to be used by the form which are first name, last name, and identification number as well as the record that contains those fields. A component is built to include those fields to present as a business unit. In this project, all the needed fields are in one table, U_MY_TEST_TBL; therefore a component is built with table, U_MY_TEST_TBL. The newly created component is now attached to a menu within PeopleSoft for user to access. The last step to be done before using that menu item is to grant permission to the end user. Permission is granted through the PeopleSoft Security Management. Figure 17 shows each item built with Application Designer in PeopleSoft followed by list of items.

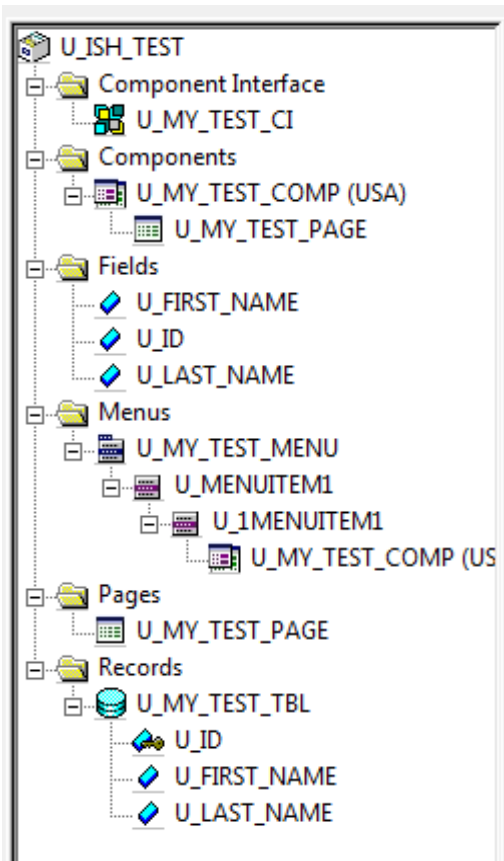


Fig: 17 Test Case Project Pane

- Project Name: U_ISH_TEST
- Field Names: U_FIRST_NAME, U_LAST_NAME, U_ID
- Record (Table) Name: U_MY_TEST_TBL

- Page Name: U_MY_TEST_PAGE
- Menu Name: U_MY_TEST_MENU
- Component Name: U_MY_TEST_COMP

Once fields, record, component, and menu are built, security is applied and tested in PeopleSoft page.

5.1.3 Building Component Interface

Once the component is thoroughly tested in PeopleSoft, the next step is to build a component interface. While creating component interface, this test case uses PeopleSoft delivered and suggested way that is by using Application Designer. Following are the generic steps for creating a component interface.

Step 1: Search and find desired component from the Application Designer menu bar.

Step 2: Once selected, right click the component and click component interface. By default all tables with records, search keys, and methods associated with component are included in the Component Interface. In this project, available methods are Create(), Cancel(), get(), Find() and Save(). In more complex components, lots of tables and fields are associated and not all of them are included in component interface by default. Those records and fields that are not included automatically can be manually added to the component interface by dragging from the component pane and dropping it into component interface pane. [The only exception to that is the search key and get key. Those key methods cannot be manually added as they are tied to the underlying business logic and PeopleCode. If desired, those can be added but underlying PeopleCode and business logic needs to be modified.]

Step 3: Once completed, the component interface is saved with a unique name so that security can be applied to it.

Step 4: Once saved, the next step is to apply PeopleSoft security for the newly created component interface so user can access for testing and further work. Security is applied through PeopleSoft Security management as described in Chapter 3 and Chapter 4

Step 5: The final step of creating component interface is to test for its functionality and accuracy. The Application Designer has built-in tools for both operations. The component interface can be tested by right clicking the component interface and click test component interface. The generated testing cases can test every method in the component interface.

Following is a snapshot of the Application Designer after building the component interface of the component, U_MY_TEST_COMP.

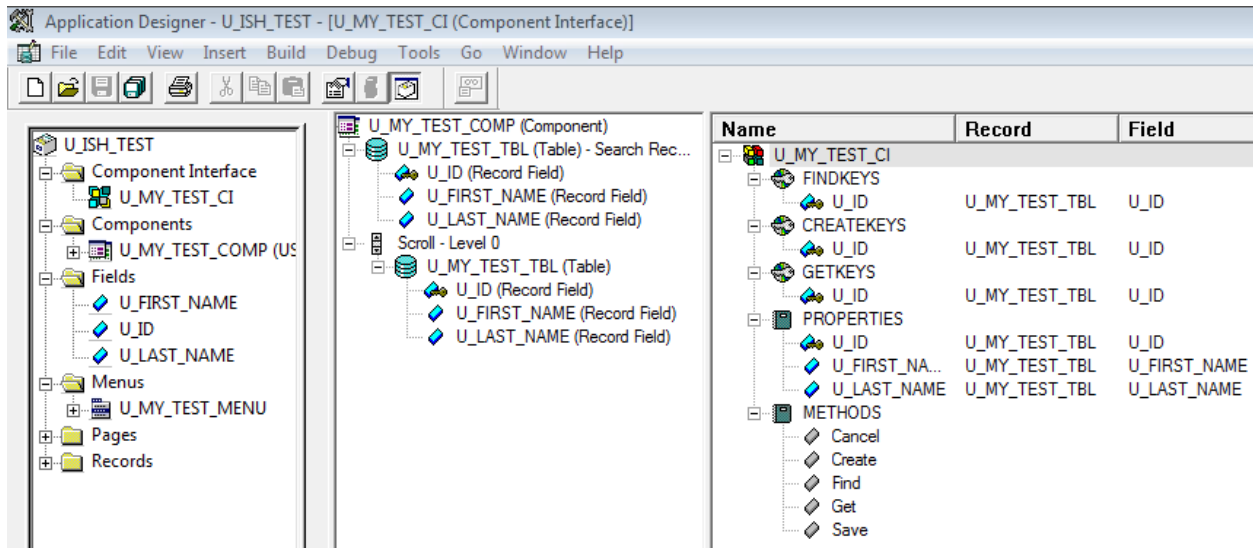


Fig: 18 Project, Component and Component Interface side by side view

5.1.4 Creating and Exposing Web Service

Once the component interface is built, a web service can be created with PeopleSoft Integration Broker, as explained in chapter 4.3.1 and 4.3.2. After applying permissions to newly created Web service, testing is done through a tool called SOAPUI and Visual Studio 2010.

5.1.5 Testing Web Service

As mentioned in Chapter 2, SOAPUI is a freeware tool for testing web services and XML files. It extracts and presents all WSDL definitions, schema definitions, operations, services, end points, WSDL contents, and other useful information based on the WSDL input.

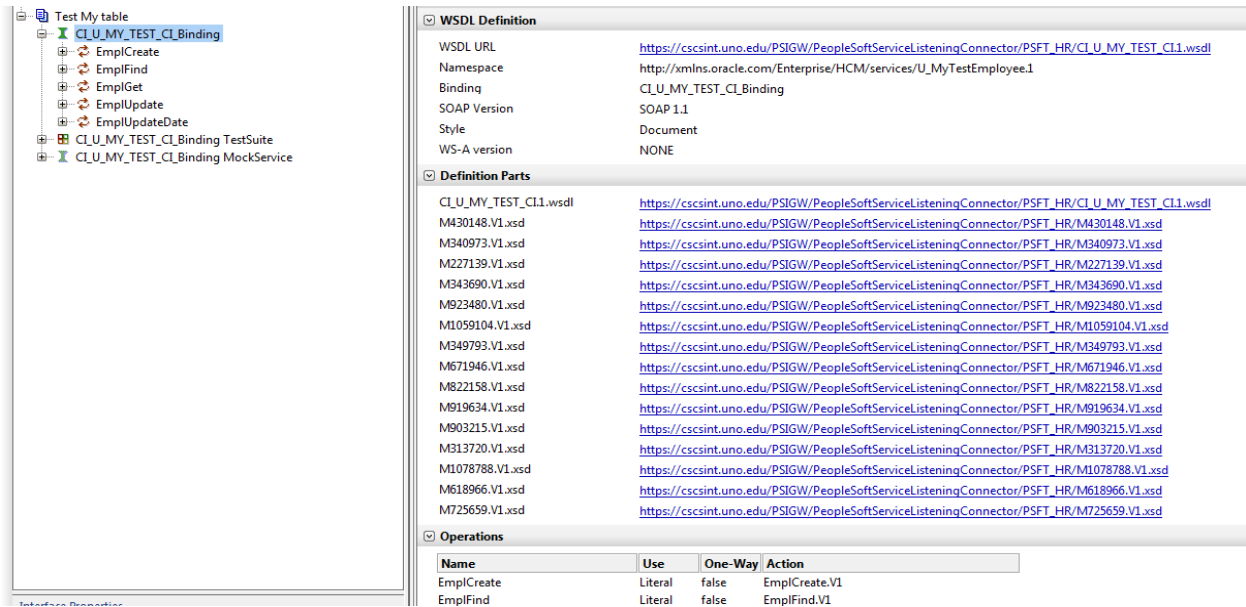


Fig: 19 SOAPUI WSDL Import

In the Figure 19, the left side shows all the methods available or exposed to external systems via PeopleSoft Web service. The right side pane shows the WSDL definition, operations and other useful information. Similarly, figure 20 shows the basic request and response soap message in SOAP UI interface.

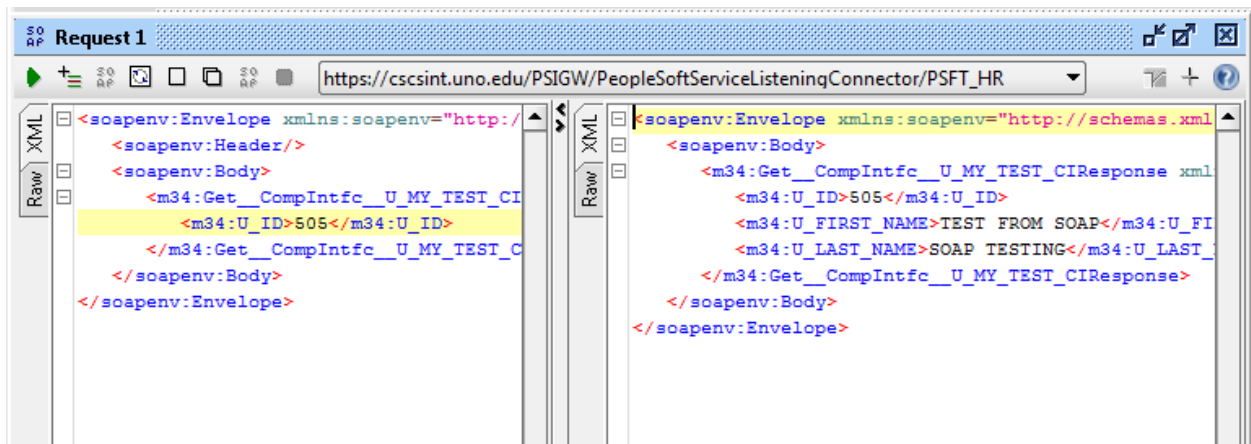


Fig: 20 Testing Person Search Method

Another set of testing was done by creating a Web application in Visual Studio 2010. A page is built with few fields and uses Consume Web Service tool in Visual Studio to create proxy server. In this test case, all five methods: Cancel (), Create (), Find (), Get (), Save () are exposed and successfully tested.

5.1.6 Implementation on InfoPath Form

Once test of SOAP and Visual Studio page is successful, the InfoPath form designer was used to design a simple demo form. InfoPath has simple form utility tool with drag-and-drop features. InfoPath was chosen for this project because of its simplicity and its compatibility with SharePoint Workflow Foundation. In this test case, update functionality of the Web service is tested. Once the design of the form is completed, it needs to be published in SharePoint Document Library as a template so it can be accessed by users over the internet via browser or InfoPath client. While uploading InfoPath form in SharePoint, form fields that are desired to identify document status and form fields desired to be used in workflow process needs to be identified. In this test case, form fields such as submitted_by, approved_by, submitted_date, approved_date and few others are identified while uploading the form as template.

Other than title, banners, button and other elements, there are three sections in the following form:

THE UNIVERSITY of NEW ORLEANS

Name Correction Request Form

Use this form to request to make correction to your name. Admissions office will review the request and make necessary changes in Student Bio/Demo Information. Please submit 2 government/state issued identification cards and social security number card (if available) along with this request.

Request ID

Employee ID

Employee ID

Name in Student Information System		Correction Requested
First Name	<input type="text" value="ARYAL"/>	<input type="text"/>
Last Name	<input type="text" value="ISHWOR"/>	<input type="text"/>

Submitted By Submitted To

Submitted Date


Approved By Approved Date

Fig: 21 InfoPath Form

First section: This part of the form is used to get student information from the PeopleSoft Campus Solution. Once the student identification number is entered, get() method is invoked to get information from database and populates in the form.

Second section: This is the section use for workflow process. Once the correct name, submitted_by, submitted_to and submitted_date is entered and the form is submitted, it is saved in SharePoint Document Library and triggers an email notification to approving department along with the location of the document in SharePoint document library. At this point, PeopleSoft Web service is not accessed as it is still in approval process.

Third section: This is the section where approval to change name takes place and PeopleSoft web service is accessed if the form is approved. After clicking to the email notification URL, the approving unit see document as below


**THE UNIVERSITY of
NEW ORLEANS**

Name Correction Request Form

Use this form to request to make correction to your name. Admissions office will review the request and make necessary changes in Student Bio/Demo Information. Please submit 2 government/state issued identification cards and social security number card (if available) along with this request.

Request ID

Employee ID

Employee ID

Name in Student Information System		Correction Requested
First Name	<input type="text" value="ARYAL"/>	<input type="text" value="Ishwor"/>
Last Name	<input type="text" value="ISHWOR"/>	<input type="text" value="Aryal"/>

Submitted By Submitted To

Submitted Date

Approved By Approved Date

Fig: 22 After Submitted

5.1.8 Applying SharePoint Workflow Foundation

Once the Submit button is clicked on the form for the first time, workflow triggers in SharePoint Workflow Foundation. Workflow in SharePoint is conditional rule that explains what needs to happen

when the form is submitted. Conditions are basic if-else statements with user defined actions. In this test, once the form is submitted for the first time, it is saved in SharePoint document library in XML format and generates a pre-formatted email to the recipient along with the absolute URL of the document. In this case, the person responsible for approving this change request form is the recipient. At this point of the process, rules are set so that no communication between the Web service client and the server takes place. Student's data is not updated in PeopleSoft until the form is approved and submitted. What follows are the workflow rules for the initial submit process.

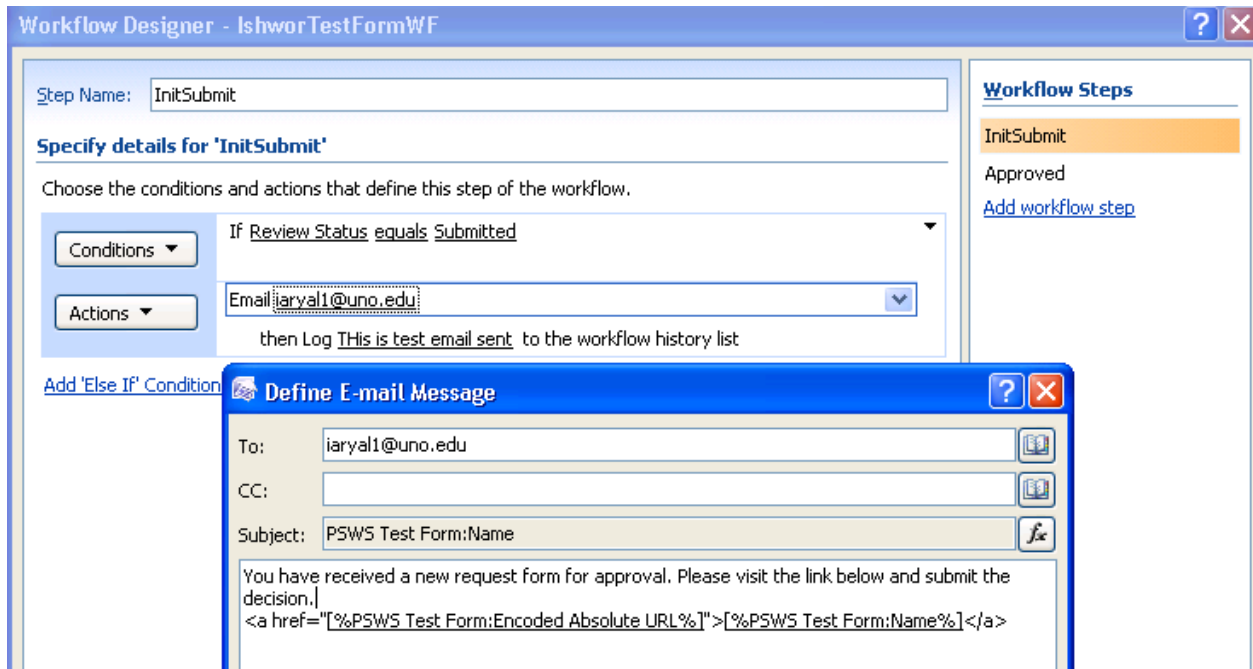


Fig: 23 Workflow Foundation in SharePoint for Form Submission

Here, when the form is submitted, SharePoint checks if the Review status is in submitted status. If it is, then a simple email is triggered. In this example, specific email address is used however in reality; this email address could be identified based on form values and route. The following diagram shows how the Document Library in SharePoint looks.

Type	Name	Modified	Modified By	Checked Out To	EMPLID	Submitted by	Submitdate	Approved by	Approved Date	Review Status	Hidden Empl	IshworTestFo
	ADM1012012-05-271.HW	5/27/2012 7:21 PM	Ishwor Aryal		101	Ishwor	5/27/2012			Submitted	101	
	REG1012012-05-25	5/25/2012 5:10 PM	Ishwor Aryal		101	ishw	5/25/2012	ish	5/25/2012	Approved	101	Completed
	ADM1012012-05-25	5/25/2012 5:02 PM	Ishwor Aryal		101	ishwr	5/25/2012	isghwor	5/25/2012	Approved	101	Completed
	1012012-05-25	5/25/2012 5:02 PM	Ishwor Aryal		101					Submitted	101	Completed

Fig: 24 SharePoint Document Library for Form Submit

Since the form is only in submitted status, fields such as Approved_by, Approved_date and workflow status are blank.

Similarly, the workflow rule for forms that are in approved status is created as show in the figure 25. If the form is finally approved and submitted, then the SharePoint checks if the form's status is approved. If this holds true, then another email is triggered to the email address who originally requested to make the name change and PeopleSoft Web service is called and update method is executed to update data in PeopleSoft Campus Solutions.

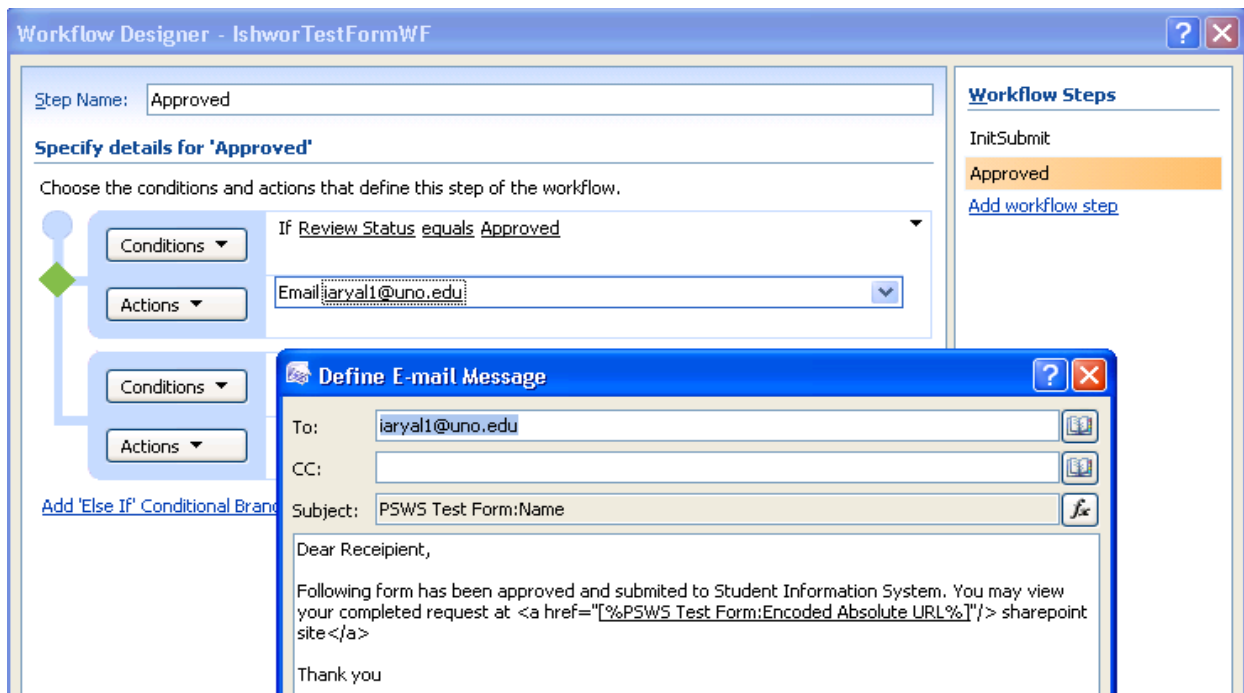


Fig: 25 Workflow Foundation in SharePoint for Form Approval

Once Form is Approved and submitted, 3 events triggered

Event 1: Save the Completed Form to SharePoint Document Library as permanent document. All fields related to approval process are filled out such as Approved BY, Date, Status and so on. Following figure highlights some of those.

Type	Name	Modified	Modified By	Checked Out To	EMPLID	Submittedby	Submitdate	Approvedby	Approved Date	Review Status	Hidden Empl	IshworTestFormWF
	ADM1022012-05-29 NEW	5/29/2012 9:26 AM	Ishwor Aryal		101	Ishwor	5/29/2012			Submitted		102
	ADM1012012-05-27	5/27/2012 7:24 PM	Ishwor Aryal		101	Ishwor	5/27/2012	ishw	5/27/2012	Approved	101	Completed
	REG1012012-05-25	5/25/2012 5:10 PM	Ishwor Aryal		101	ishw	5/25/2012	ish	5/25/2012	Approved		Completed

Fig: 26 SharePoint Document Library view after Form Approved

Event 2: Email notification is triggered to person who initially submitted notifying the approval status.

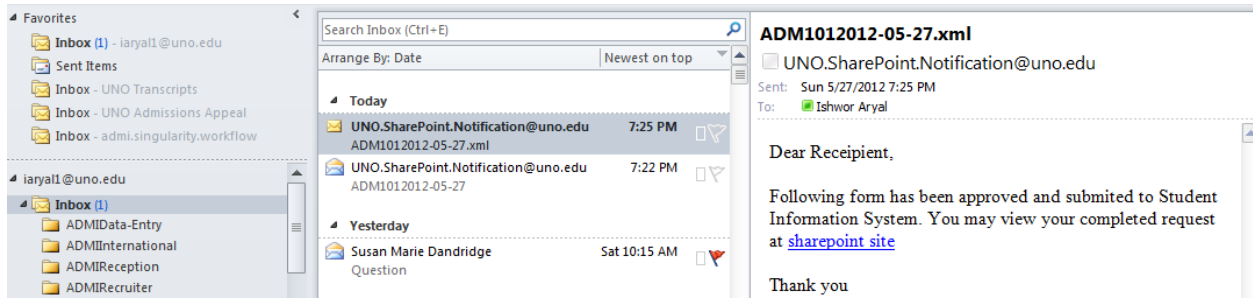


Fig: 27 Email Triggered after Form is approved

Event 3: Appropriate action is taken in PeopleSoft based on the method used from Web service. In this test case, update method is used to correct name. Following figure shows the change in PeopleSoft.

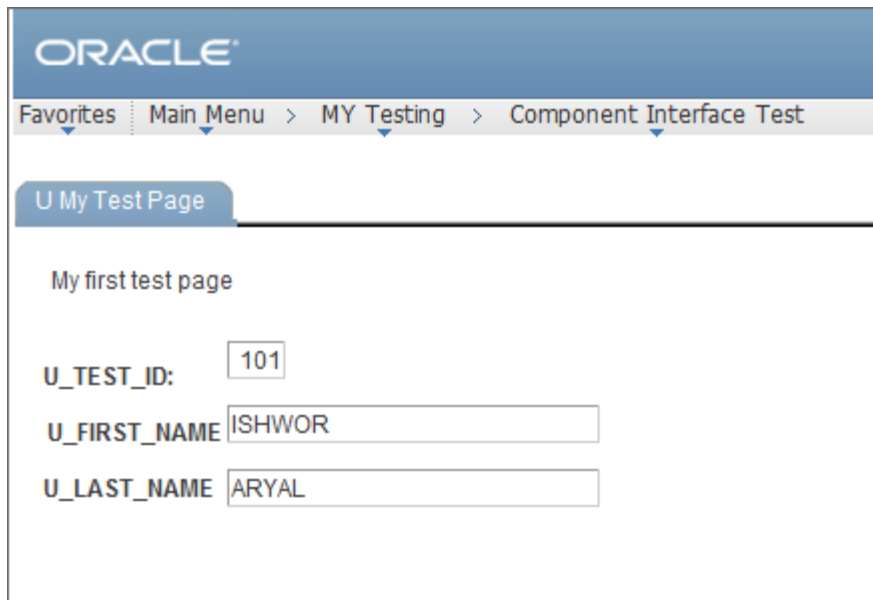


Fig: 28 PeopleSoft Data Change View after Form is approved

5.2 Case II – Retire or Rehire Employee in PeopleSoft Human Resource Component

5.2.1 Problem Description

In this test case, Retire or Rehire request is processed. A request is made to the human resource staff by a business manager to take action for an employee. Once a change request form is submitted, the form is routed to the supervisor or department head of that employee for approval via email triggered by predefined workflow in SharePoint. A copy of the original form is stored in a secure location. If the form is approved, the employee's status is changed in PeopleSoft Campus Solution by invoking update method using the Web service. A copy of the approved form is stored in SharePoint Document Library. If it is denied, the denied form is still stored in document library but employee data is not updated in PeopleSoft Campus Solution.

In this test case, PeopleSoft delivered component interface, CI_JOB_DATA is chosen to build Web service. The component chosen for this test incorporates job related employee information in a single component, JOB_DATA.

5.2.2 Configuration

As discussed in chapter 3 and chapter 4, configuration includes access privilege for the component interface and underlying component for the end user. Since the component and component interface being used in this test case is delivered by PeopleSoft Campus Solution, no further action is required other than testing available methods for the component interface. Component Interface, CI_JOB_DATA offers four methods namely: Cancel, Find, Get and Save. This component interface does not provide create method which prevents external application from adding new employee in the system. PeopleSoft has delivered another component called CI_JOB_DADA_EMPL for creating or adding new employee in the system which is not used in this project.

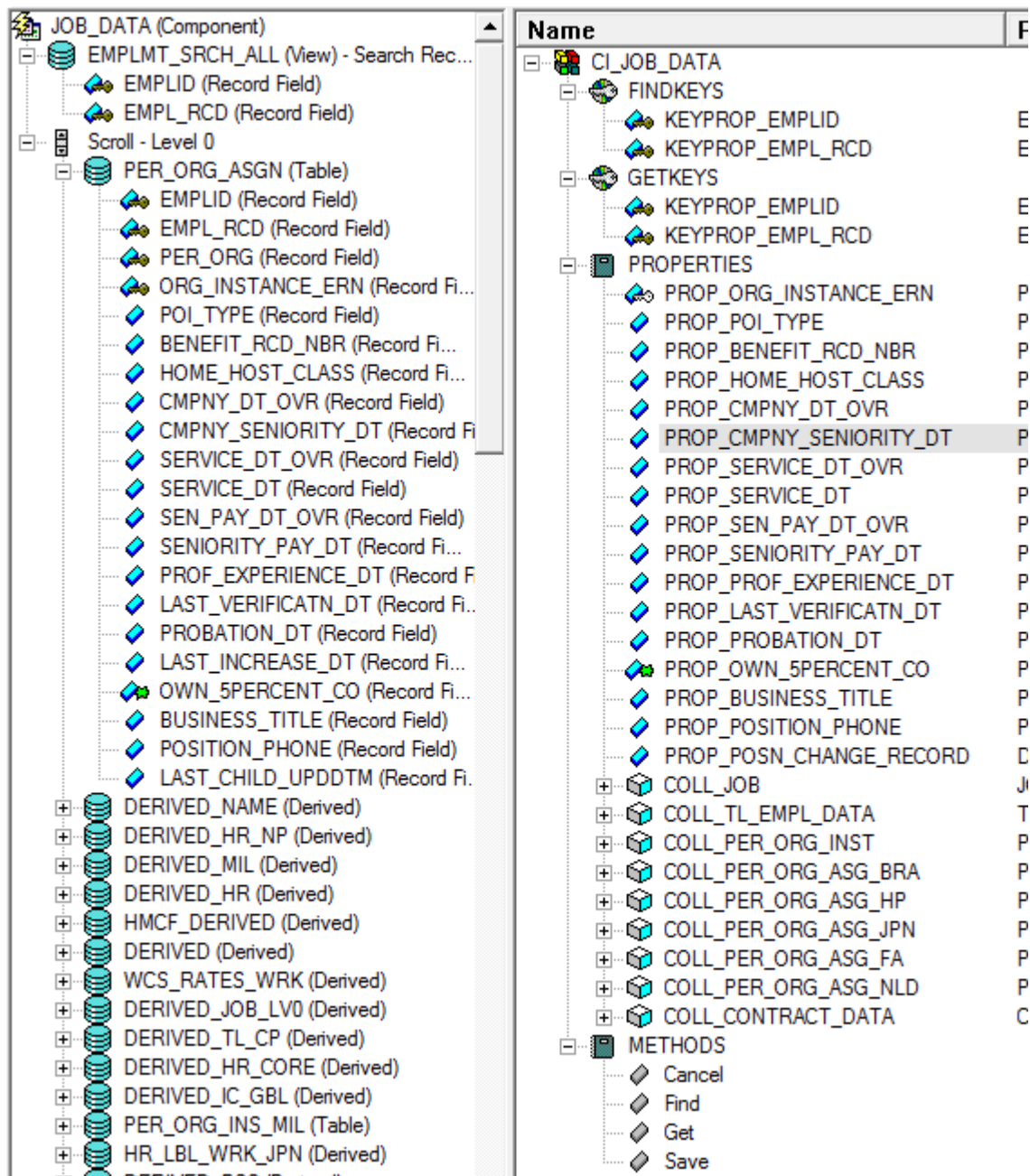


Fig: 29 Component and Component Interface for Job Data

In Figure 29, left pane shows the component along with all the records associated with that. The right pane shows the component interface along with keys, properties and methods available. Closer look at the right pane indicates that the component interface has a collection of records associated with the underlying component.

5.2.4 Building and Exposing Web Service

Once the component interface is thoroughly tested in PeopleSoft Application Designer, the next step is to build a Web service as explained in chapter 4.3.1 and 4.3.2. After applying access privilege to newly created Web service, test is performed through a tool called SOAPUI. Section 5.2.5 explains the steps taken to test the Web Service.

5.2.5 Testing Web Service

As we discussed and tested in chapter 5, case I, first test is done using SOAPUI. In the figure 30, the left side shows all the methods available or exposed to the external systems via PeopleSoft Web service for Job data component. The right side pane shows the WSDL definition, WSDL contents, operations and other useful information.

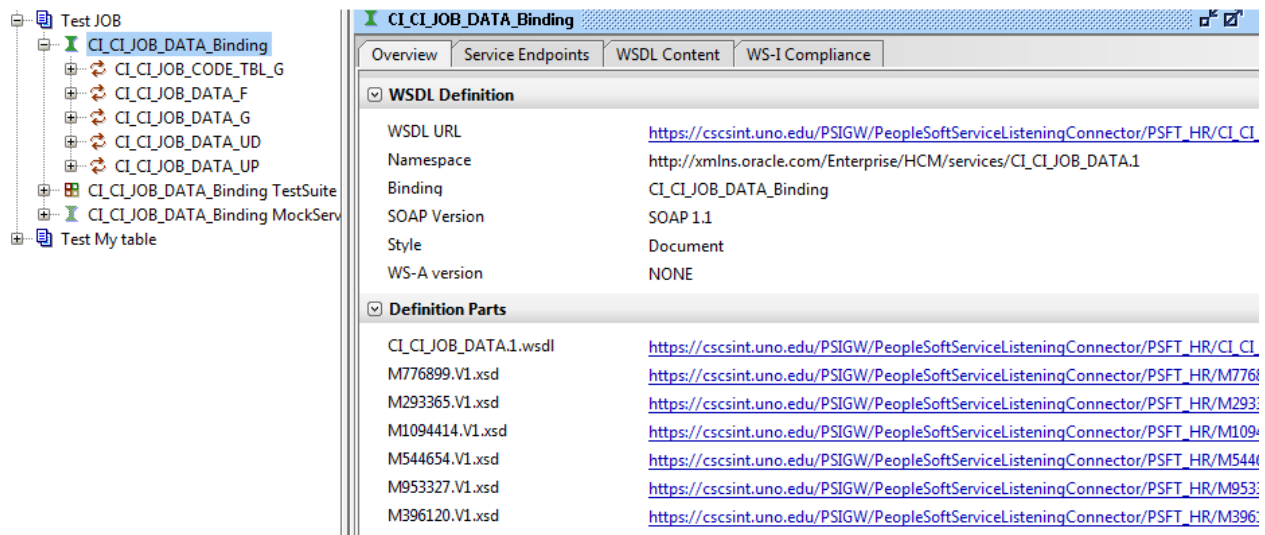


Fig: 30 SOAP WSDL Import

In left pane, there are two types of update methods exposed by the Web service; update and updateData. UpdateData method is used to modify existing data which mean that the historic data is not maintained whereas update method preserves historic data by not modifying the existing data instead it inserts a new row of data with required change. In this project, update method is tested.

The Figure 31 shows snapshot of input parameters and output data after executing get() in SOAPUI. The left pane is the input parameters where employee identification number and record number are use. The right pane shows the result with lots of information. In this test case we are interested in

employee status and action taken for the employee. In this example, employee number 0045 is still an active employee with Rehire as status.

```

Raw XML
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header/>
  <soapenv:Body>
    <m92:Get__CompIntfc_CI_JOB_DATA>
      <m92:KEYPROP_EMPLID>0045</m92:KEYPROP_EMPLID>
      <m92:KEYPROP_EMPL_RCD>0</m92:KEYPROP_EMPL_RCD>
    </m92:Get__CompIntfc_CI_JOB_DATA>
  </soapenv:Body>
</soapenv:Envelope>

Raw XML
<m92:PROP_HR_STATUS>A</m92:PROP_HR_STATUS>
<m92:PROP_APPT_TYPE>0</m92:PROP_APPT_TYPE>
<m92:PROP_MAIN_APPT_NUM_JPN>0</m92:PROP_MAIN_APPT_NUM_JPN>
<m92:PROP_POSITION_OVERRIDE>Y</m92:PROP_POSITION_OVERRIDE>
<m92:PROP_POSN_CHANGE_RECORD>N</m92:PROP_POSN_CHANGE_RECORD>
<m92:PROP_EMPL_STATUS>A</m92:PROP_EMPL_STATUS>
<m92:PROP_ACTION>REHS</m92:PROP_ACTION>
<m92:PROP_ACTION_DT>2012-02-26</m92:PROP_ACTION_DT>
<m92:PROP_ACTION_REASON/>
<m92:PROP_LOCATION>KUNY00</m92:PROP_LOCATION>
<m92:PROP_TAX_LOCATION_CD>KUA200</m92:PROP_TAX_LOCATION_CD>
<m92:PROP_JOB_ENTRY_DT>2012-02-26</m92:PROP_JOB_ENTRY_DT>
  
```

Fig: 31 Get() method of Web Service and Result in SOAPUI

The Figure 32 shows the PeopleSoft User Interface view for the status of the employee.

Andy Benoit	EMP	ID:	0045	Empl Rcd #:	0
Work Location Find First 1 of 1 Last					
HR Status:	Active	Payroll Status:	Active	<input type="button" value="Go To Row"/> <input type="button" value="+"/> <input type="button" value="-"/>	
*Effective Date:	02/26/2012	Sequence:	0	*Job Indicator:	Primary Job
*Action:	Rehire	Reason:			
Current					
Last Start Date:	02/26/2012	Termination Date:			
Expected Job End Date		Position Entry Date:	02/26/2012		

Fig: 32 Status of employee: Rehired and Active since 02/26/2012

Figure 35 shows the SOAPUI test on update method exposed by JOB_DAT Web service. In the left pane, input parameters are set. All elements not requiring update must be deleted in order for update to execute. Unused elements carry empty strings that cause PeopleCode to generate errors based on the business logic for that component. Value of 1 highlighted on the right side of the figure 35 indicates that the update was successful. In this example, we are attempting to retire an employee, 0045 effective on 04/15/2012.

```

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header/>
  <soapenv:Body>
    <m10:Update__CompIntfc_CI_JOB_DATAResponse xmlns:m10="http://schemas.xmlsoap.org/soap/envelope/">
      <m10:notification>1</m10:notification>
      <m10:detail>
        <m10:messages>
          <m10:type>Warning</m10:type>
          <m10:messagesetnumber>15</m10:messagesetnumber>
          <m10:messagenumber>9</m10:messagenumber>
          <m10:mesagetext>Warning -- date out of range. (15,9)
          <m10:explaintext>The date entered is either more than
        </m10:messages>
      </m10:detail>
    </m10:Update__CompIntfc_CI_JOB_DATAResponse>
  </soapenv:Body>
</soapenv:Envelope>

```

Fig: 33 Testing of Update method in SOAPUI

In figure 34, immediate reflection of the change is shown via PeopleSoft Campus Solution User Interface page. Noticeably, new row is added on top of the existing one with the most recent date (04/15/2012) with the status of employment “Inactive” and the action taken is “Retired”. This update method preserves the historic data by not modifying the existing data. In other word, update works as insert where data is inserted into the table(s). The only difference is that insert needs all the required fields as parameter whereas update needs only key values and fields that need update. As we can see in figure 34, all data are still inserted using Update method which are retrieved in runtime and stored in memory for processing. Each row can be viewed by using the arrow key on the top right corner of the page.

Andy Benoit	EMP	ID: 0045	Empl Rcd #: 0
<div style="text-align: right;">Find First 1 of 2 Last</div>			
HR Status:	Inactive	Payroll Status:	Retired
*Effective Date:	04/15/2012	Sequence:	1
*Action:	Retirement	*Job Indicator:	Primary Job
Reason: <input type="text"/>			
Current			
Last Start Date:	02/26/2012	Termination Date:	04/14/2012
Expected Job End Date	<input type="text"/>		

Fig: 34 PeopleSoft User Interface of employment data after update method is executed

If the same update was done using UpdateData method exposed by Web service, there would have been only one record for this employee and we would have lost historic data for that employee; such as hired date, hired reason and other relevant information. UpdateData method is used rarely in business unit and very few users are granted privilege to do so.

5.2.6 Implementation

Once Web service is thoroughly tested in SOAPUI, implementation in the InfoPath form is one of biggest challenge for this test case. This is because of the complexity of the Web Service WSDL file, schema and definition provided by PeopleSoft. Testing of the same Web service is successfully done in SOAPUI however implementing the same functionality is hard in InfoPath for the following reasons.

- PeopleSoft Web service has combinations of simple and complex object types. This means that a complex object has multiple simple objects.
- As discussed in section 5.2.5, while executing update method, elements that do not require updating must be deleted. InfoPath does not allow developer to strip elements on the Form so update cannot be done
- When using web service in InfoPath, it can execute get data method from the PeopleSoft Web service and display data from the complex object however it could not expand the complex object where developer can attach data value within the simple object of that complex object. For example, to terminate employee, following information is required as input parameter:
 - Employee number (KEYPROP_EMPLID),
 - Record number (KEYPROP_EMPL_RCD),
 - Date (KEYPROP_EFFDT),
 - Sequence (KEYPROP_EFFSEQ)
 - Action (PROP_ACTION)

The figure 5.2.4 shows the structure of the soap request; employee number and record are simple object type whereas date, sequence and action are simple type within a complex object type COLL_JOB.

When creating InfoPath form, developer can see only complex object type, COLL_JOB which is not sufficient to pass required simple object type as update parameters.

5.2.7 Alternative Proposed Solution

After extensive study, couple solutions are discovered and proposed as a workaround which will be discussed in this chapter but implementation is set for future work. Both solutions requires extensive programming knowledge and resources to implement in SharePoint Workflow Foundation. The first solution proposed is to write custom code within InfoPath form's submit method to update employee data. Another is to create standard .net Web service methods (Get, Find, Create, Update, Updatedata). Each method should have the parameters that allow the input/output data. Then in the method's logic,

instantiate the PeopleSoft's Web service to assign or get the values to or from the PeopleSoft Web service's properties. InfoPath form then can reference .NET Web service that consumes the underline PeopleSoft Web service. Since the objective of this project is to come up with the solution to minimize programming resource, maximize version compatible and minimize maintenance, neither of the solutions are adapted in this project. However; to show the possibility, simple test is done using .NET environment to consume PeopleSoft Web service methods without tying it to SharePoint Workflow Foundation.

Chapter 6 Conclusion and Future Work

In this project work, component and Component Interface are studied and generated Web service based on selected component Interface and exposed to external application. We studies the techniques of integrating PeopleSoft Campus Solution product with non-PeopleSoft application using emerging technology called Web service.

Specifically, we demonstrated the use of Component Interface Based Web Service in InfoPath form to be integrated with SharePoint Document Library and Workflow Foundation. The idea being simple to use yet robust technique to expose PeopleSoft business logic without the need of extensive programming and modification of delivered code is tested and achieved in this project work.

In this project work, commonly used technologies such as Web service, SOAP, SOAPUI, Visual Studio, InfoPath Designer, SharePoint Server, SharePoint Designer, SharePoint Workflow Foundation, and SharePoint Document Library to plan, design, create, test and implement; to integrate PeopleSoft Campus Solution to external application.

This same technique can be utilized in several different areas such as mobile application for student/faculty self-service, integrate external Online Application for Admissions to PeopleSoft, single sign on in SharePoint sites, or even expose relevant PeopleSoft components/menus/pages/function to SharePoint without compromising security and business logic.

References

- [1] J. Dean, A. Gravel (Eds.), *COTS-Based Software Systems, (Proceedings of the First International Conference on COTS-Based Software Systems (ICCBSS 2002)*, Orlando, FL, USA, February 4-6, 2002), LNCS 2255, Springer-Verlag, Heidelberg, Germany, 2002.
- [2] D.A. Fisher, B.C. Meyers and P. Place, “Conditions for Achieving Network-Centric Operations in systems of Systems”, Carnegie Mellon Software Engineering Institute technical note, CMU/SEI-2007-003, 2007.
URL: <http://www.sei.cmu.edu/reports/07tn003.pdf>
- [3] S. Tu, G. Li and P. Augustin, “Strategies for Integration of a non-OO EIS and the J2EE Framework”, proceedings of the 26th Computer Software and Applications Conference (CompSAC) , pp 246-251, Oxford, England, August 26-29, 2002.
- [4] D. Luzeaux, J.R. Ruault, J.L. Wippler, *Complex System and Systems of Systems Engineering*, ISTE Ltd and John Wiley & Sons Inc, 2011
- [5] Oracle Corporations, Oracle PeopleTools Campus Solution Release 9.0 (CS) up to and including Bundle 24/AF, 2012
URL: http://docs.oracle.com/cd/E29376_01/hracs90r5/eng/psbooks/index.htm
- [6] Enterprise PeopleTools 8.50 PeopleBook: PeopleSoft Component Interfaces, PeopleSoft Application Designer Developer's Guide, Oracle Corporations, 2012
- [7] Oracle Corporation, Oracle PeopleSoft Internet Architecture and Web Services - A Technology Overview – PeopleSoft, Technology Whitepaper, March 2002
- [8] Oracle Corporation, the Out of the Box Application Sample Application Kit for PeopleSoft: Application Overview & Installation
- [9] Ali Khawaja Microsoft Consulting Services - Office SharePoint Server 2007 and PeopleSoft Integration, Whitepaper, Published: 5/25/2007
- [10] Abhishek Nigam, Teekam Chand Goyal (Intelligroup, Inc), Connecting to PeopleSoft® from Microsoft® Office InfoPath Whitepaper Prepared by: Published: September 2004
- [11] David Sohigian, PeopleSoft and Babar Batla, Connecting to PeopleSoft® Web Services with Microsoft® Visual Studio.Net ® Microsoft Published: September 2004
- [12] The University of New Orleans, SharePoint Intranet Site
URL: <https://privateers.uno.edu:44300/students/admissions/default.aspx>
- [13] Michele Leroux Bustamante, Microsoft SharePoint Workflow Foundation, IDesign, May 2008
URL: <http://msdn.microsoft.com/en-us/library/cc748597.aspx>
- [14] Oracle Corporations, PeopleCode Training Manual - PeopleTool 7.5, June 2007
- [15] Bill Sempf, Donald Xie, James Greenwood, Professional Visual Studio .NET, Apress 2004

[16] Brian Benz with John R. Durant, XML Programming Bible, Wiley Publishing, Inc. 2003
Anderson, Lynn (2001). Understanding PeopleSoft8. Sybex. pp. 22. ISBN 0-7821-2930-7

VITA

Ishwor Aryal was born in Kathmandu, Nepal. He earned his Bachelor of Science from Colorado Technical University with Computer Science as major on March 2006. He was enrolled in the graduate program in Computer Science at the University of New Orleans in January 2007.