ORACLE®

JD EDWARDS WORLD

JD Edwards World

Service Enablement Guide

Version A9.1

Revised – October 2, 2007

JD Edwards World

Open Source Disclosure

Oracle takes no responsibility for its use or distribution of any open source or shareware software or documentation and disclaims any and all liability or damages resulting from use of said software or documentation. The following open source software may be used in Oracle's PeopleSoft products and the following disclaimers are provided.

Table of Contents

Introduction 2

Overview of JD Edwards World Service Enablement 2
Components of JD Edwards World Service Enablement 4
Installed Web Services 4

How to Turn Service Enablement on 13

Starting the Transidiom Gateway and AIS 13
Deploying and Enabling Web Services 15
Deploying to Oracle Application Server 15
Deploying to WebSphere Application Server 19
Testing Web Services 24
Testing on Oracle Application Server 25
Extracting WSDL Files for Testing with WebSphere Application Server 29

Components of a JD Edwards World Web Service 32

Transidiom Components 32
Java Layer 35
Retrofitting Web Services 36

Appendix A – Processing Options 37

Processing Options 37

Introduction

Overview of JD Edwards World Service Enablement

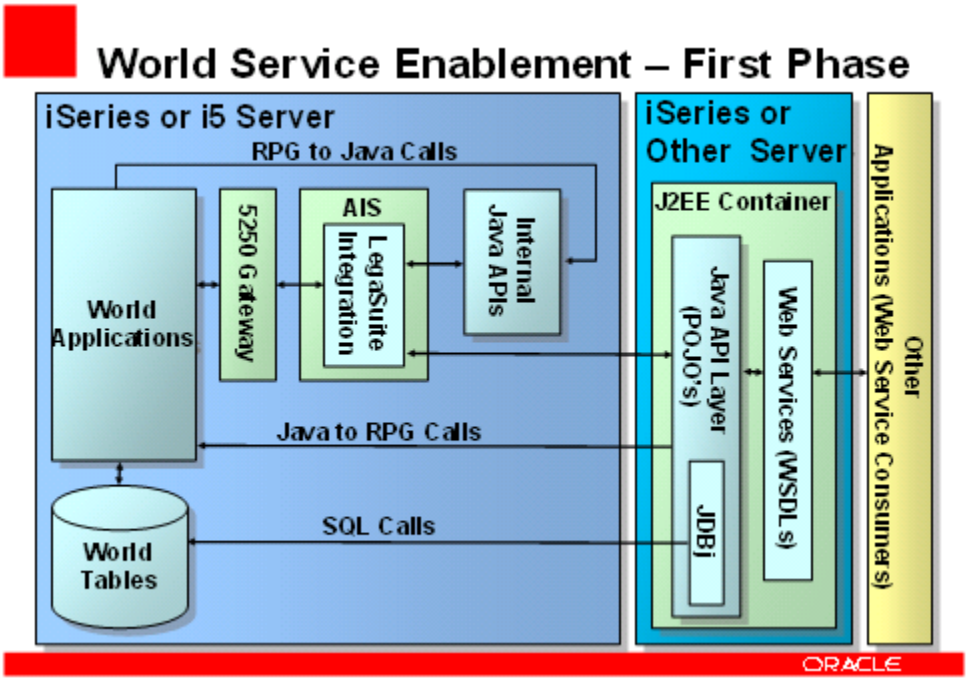
The main objective of the JD Edwards World Service Enablement product is to expose JD Edwards World business process to the outside world. To do this task, a Java API layer was created and is being exposed to outside applications through Web Services.

To communicate with JD Edwards World applications, the API layer uses one of several options.

- Option One - Use Seagull's LegaSuite Integration (formerly Transidiom) product to access the JD Edwards World interactive applications through scripting scenarios. The data from the interactive applications will be passed from JD Edwards World to the API layer modules.
- Option Two - Have the API layer make RPG program calls from the Java code using the JT400 toolset.
- Option Three - Used to perform SQL calls from the Java code to the JD Edwards World database tables. The JD Edwards World JDBj API is used to process the SQL, as it allows for data conversion from the JD Edwards World database. The communication option used depends on the type of business process and could include more than one of the choices.

Data objects were created for each of the business processes. The data objects are exposed to outside applications by utilizing Web Services (that use a WSDL interface) for each business process. The Web Services are bundled inside an Enterprise Archive file, or .ear file, which can be deployed to a Java application server. The Web Services are J2EE compliant and can be used, or consumed, by other Java programs. The WSDL interface also allows for the Web Services to be consumed by third party integration applications, such as Oracle's BPEL Process Manager or IBM's WebSphere Business Integrator.

The following diagram gives a graphical view of the JD Edwards World Service Enablement architecture.



The different colors in the diagram represent different components of Service Enablement.

- Light blue – JD Edwards World components
- Light green – Third party components
- Yellow – Non-JD Edwards World applications

The following explanations the terms used in the diagram:

- 5250 Gateway – part of the Seagull Transidiom connector for reading System i communication stream
- AIS – Seagull Application Integration Server
- API – Application Programming Interface
- J2EE – Java 2 Platform, Enterprise Edition
- J2EE Container – J2EE application server (environment)
- Java – programming language designed for distributed environments
- JDBj – Oracle custom JDBC database connector, includes JD Edwards data scrubbing
- RPG – programming language JD Edwards World is written in
- SQL – Structured Query Language
- Transidiom – Seagull Software product for invoking System i applications
- WSDLs – Web Service Definition Language (web service document)
- What's Installed

Components of JD Edwards World Service Enablement

Once you complete the install of JD Edwards World Service Enablement, the following components should be installed:

> **Note:** There are references to Seagull's LegaSuite Integration as Transidiom, which coincides with the version of the software we are utilizing.

- Transidiom Central on the local machine. Transidiom Central is the administrative tool for the Transidiom suite. It allows administrators to configure and remotely administer the Application Interface Server (AIS).
- Transidiom Gateway on either the local machine or the System i. The Transidiom Gateway provides a communication layer between the JD Edwards World applications on the System i and the Application Interface Server (AIS).
- Transidiom Application Interface Server (AIS) on either the local machine or the System i. The AIS is a runtime server that processes a host application (JD Edwards World) request encoded in XML, invokes the host application, and returns the application's output as a response encoded in XML.
- JD Edwards World A9.1 Service Enablement Transidiom scripts deployed on the either the local machine or the System i. These are JD Edwards World specific scripts that are used to run interactive application sessions used to process data for the web services.
- JD Edwards World A9.1 Service Enablement Java application .ear file for either the Oracle Application Server (OAS) or the WebSphere Application Server (WAS). For OAS, the file is WorldServicesEnablement-A9.1-0.ear. For WAS, the file is WorldSOA_A91_WebsphereEAR.ear. This file contains the Java API .jar file, the web services files, the web services configuration file (config.xml) and all the supporting files. The file can be deployed to the application server so that the web services can be run.
- JD Edwards World A9.1 Service Enablement Java API .jar file, along with supporting .jar files, is on the System i in the WORLDSOA_JAVA directory under the Root IFS directory. The system uses this in conjunction with the A9.1 Approvals Management system.
- Source code for the services is in a Source folder in the installation working folder. Source code is in the form of .zip files for use in Oracle's JDeveloper or IBM's WebSphere Development Studio Client (WDSC). Also available is source for the Service Enablement Transidiom scripts.

Installed Web Services

The following lists Web Services installed with JD Edwards World Service Enablement.

Private methods Information and other details can be found in Classes' Javadocs.

Foundation

- Web Service: Get next number and create a new batch control record
  - Package: utilities
  - WSDL: BatchControl.wsdl
  - Implementation Class: BatchCtrlImpl.java
  - Public Operations / Methods: createBactchCtrl()
  - Implemented Via: RPG Service Program X0097
- Web Service: Business Unit Scrub
  - Package: utilities
  - WSDL: BusinessUnitMaster.wsdl
  - Implementation Class: BusinessUnitMasterImpl.java
  - Public Operations / Methods: getBusinessUnit()
  - Implemented Via: RPG Service Program X0006
- Web Service: Date Values Scrub / Conversion
  - Package: utilities
  - WSDL: ConvertDates.wsdl
  - Implementation Class: ConvertDatesImpl.java
  - Public Operations / Methods: convertDates()
  - Implemented Via: RPG Service Program X0028
- Web Service: Currency Conversion
  - Package: utilities
  - WSDL: CurrencyConversion.wsdl
  - Implementation Class: CurrencyConversionImpl.java
  - Public Operations / Methods: convertCurrencyValue()
  - Implemented Via: RPG Service Program X00195
- Web Service: Get Next Number
  - Package: utilities
  - WSDL: NextNumber.wsdl
  - Implementation Class: NextNumberImpl.java
  - Public Operations / Methods: getNextNumber()
  - Implemented Via: RPG Service Program X0010
- Web Service: User Defined Code values Inquiry
  - Package: utilities
  - WSDL: UserDefinedCodes.wsdl
  - Implementation Class: UserDefinedCodesImpl.java

        - Public Operations / Methods: getUserDefinedCode()
        - Implemented Via: RPG Service Program X0005
    - Web Service: Bank Account Reference Inquiry / Update / Delete
        - Package: utilities
        - WSDL: BankAccount.wsdl
        - Implementation Class: BankAccountImpl.java
        - Public Operations / Methods: getBankAccounts(), setBankAccounts(), deleteAccounts()
        - Implemented Via: Transidiom over screens V0030, V0030W1
    - Web Service: Date Format for output
        - Package: utilities
        - WSDL: FormatData.wsdl
        - Implementation Class: FormatDataImpl.java
        - Public Operations / Methods: formatData()
        - Implemented Via: RPG Service Program X00162

Address Book

- Web Service: Address Book Number Validation
    - Package: utilities
    - WSDL: AddressBookMaster.wsdl
    - Implementation Class: AddressBookMasterImpl.java
    - Public Operations / Methods: getAdressBookMaster()
    - Implemented Via: RPG Service Program X0101
- Web Service: Address Book Revisions Inquiry / Add / Update / Delete
    - Package: utilities
    - WSDL: AddressBook.wsdl
    - Implementation Class: AddressBookImpl.java
    - Public Operations / Methods: getAddressBook(), setAddressBook(), deleteAddressBook(), addAddressBook (), updateAddressBook(), setControlFields()
    - Implemented Via: Transidiom over screens V01051, V010513, V010512, V0111, V0111W
- Web Service: Customer Master Inquiry / Add / Update / Delete
    - Package: utilities
    - WSDL: Customer.wsdl
    - Implementation Class: CustomerImpl.java

        - Public Operations / Methods: getShortCustomer(), setShortCustomer(), getCustomer(), setCustomer(), deleteCustomer(),
        - Implemented Via: Transidiom over screens V01053, V4206, V42061
- Web Service: Socio-Economic Information Inquiry / Add / Update / Delete
    - Package: utilities
    - WSDL: Demographics.wsdl
    - Implementation Class: DemographicsImpl.java
    - Public Operations / Methods: getDemographics(), deleteDemographics(), setDemographics()
    - Implemented Via: Transidiom over screen V010514
- Web Service: Email / URL Revision Add / Update / Delete
    - Package: utilities
    - WSDL: EmailAddress.wsdl
    - Implementation Class: EmailAddressImpl.java
    - Public Operations / Methods: setEmailAddress(), deleteEmailAddress()
    - Implemented Via: Transidiom over screen V01018
- Web Service: Related Addresses Inquiry / Update / Delete
    - Package: utilities
    - WSDL: RelatedAddress.wsdl
    - Implementation Class: RelatedAddressImpl.java
    - Public Operations / Methods: getRelatedAddresses(), setRelatedAddresses(), deleteRelatedAddress()
    - Implemented Via: Transidiom over screen V01017
- Web Service: Supplier Master Inquiry / Add / Update / Delete
    - Package: utilities
    - WSDL: Supplier.wsdl
    - Implementation Class: SupplierImpl.java
    - Public Operations / Methods: getSupplier(), setSupplier(), deleteSupplier()
    - Implemented Via: Transidiom over screens V01054, V4306

Accounts Receivable

- Web Service: Payment Terms Inquiry
    - Package: financials
    - WSDL: PaymentTerms.wsdl
    - Implementation Class: PaymentTermsImpl.java

- Public Operations / Methods: getPaymentTerms()
- Implemented Via: RPG Service Program X0014

Payroll

- Web Service: Employee Master Inquiry
  - Package: utilities
  - WSDL: EmployeeMaster.wsdl
  - Implementation Class: EmployeeMasterImpl.java
  - Public Operations / Methods: getEmployeeMaster(), getEmployeeMasterLimited(), getEmployeeMasterBasic(), getEmployeeMasterRecharge()
  - Implemented Via: RPG Service Program X0601

General Accounting

- Web Service: Journal Entry Add / Update / Inquiry
  - Package: financials
  - WSDL: JournalEntry.wsdl
  - Implementation Class: JournalEntryImpl.java
  - Public Operations / Methods: getJournalEntry(), setJournalEntry()
  - Implemented Via: JDBJ
- Web Service: Account Master Information Inquiry / Validation
  - Package: utilities
  - WSDL: AccountValidation.wsdl
  - Implementation Class: AccountValidationImpl.java
  - Public Operations / Methods: performAccountValidation()
  - Implemented Via: RPG Service Program X0901
- Web Service: G / L date Inquiry / Edit
  - Package: utilities
  - WSDL: GLDate.wsdl
  - Implementation Class: GLDateImpl.java
  - Public Operations / Methods: getPeriodFYOrGLDate(), getLastDateOfPeriodForGLDate()
  - Implemented Via: RPG Service Program X09031
- Web Service: Subledger and Subledger Type Inquiry / validation
  - Package: utilities
  - WSDL: SubledgerEdit.wsdl

    - Implementation Class: SubledgerEditImpl.java
    - Public Operations / Methods: performSubledgerEdit()
    - Implemented Via: RPG Service Program X0909
- Web Service: GL Revise Single Account Inquiry / Add / Update
    - Package: utilities
    - WSDL: GLReviseAccount.wsdl
    - Implementation Class: GLReviseSingleAccountImpl.java
    - Public Operations / Methods: setGLAccount(), getGLAccount()
    - Implemented Via: Transidiom over screen V09011

Fixed Assets

- Web Service: Fixed Assets Cost Summary Inquiry
    - Package: utilities
    - WSDL: FACostSummary.wsdl
    - Implementation Class: FACostSummaryImpl.java
    - Public Operations / Methods: getCostSummary()
    - Implemented Via: Transidiom over screen V122101
- Web Service: Fixed Assets Master Information Inquiry / Add / Update
    - Package: utilities
    - WSDL: FAMasterInformation.wsdl
    - Implementation Class: FAMasterInformationImpl.java
    - Public Operations / Methods: setFAMaster(), getFAMaster()
    - Implemented Via: Transidiom over screens V1201, V12PAR, V12STAT, V12LOC, V12UPD

Manufacturing

- Web Service: Distribution / Manufacturing AAI's Inquiry
    - Package: utilities
    - WSDL: ManufacturingDistributionAAI.wsdl
    - Implementation Class: ManufDistAAIImpl.java
    - Public Operations / Methods: getManufDistAAI()
    - Implemented Via: RPG Service Program X4090

Inventory

- Web Service: Item Master Information Inquiry

        - Package: inventory
        - WSDL: ItemMaster.wsdl
        - Implementation Class: ItemMasterImpl.java
        - Public Operations / Methods: getItemMaster()
        - Implemented Via: RPG Service Program X4101
    - Web Service: Bulk Unit of Measure Conversion
        - Package: inventory
        - WSDL: UnitOfMeasureBulkConversion.wsdl
        - Implementation Class: UMBulkConversionImpl.java
        - Public Operations / Methods: convertUMBulk()
        - Implemented Via: RPG Service Program X41251
    - Web Service: Unit of Measure Conversion
        - Package: inventory
        - WSDL: UnitOfMeasureConversion.wsdl
        - Implementation Class: UMConversionImpl.java
        - Public Operations / Methods: getQuantityConverted()
        - Implemented Via: RPG Service Program X41003
    - Web Service: Item Unit of Measure Conversion
        - Package: inventory
        - WSDL: UnitOfMeasureItemConversion.wsdl
        - Implementation Class: UMItemConversionImpl.java
        - Public Operations / Methods: convertUMItem()
        - Implemented Via: RPG Service Program X41002
    - Web Service: Sales Order Summary Availability Inquiry
        - Package: utilities
        - WSDL: SummaryOfAvailability.wsdl
        - Implementation Class: SummaryOfAvailabilityImpl.java
        - Public Operations / Methods: getAvailability()
        - Implemented Via: Transidiom over screen V41202

Sales Order

- Web Service: Customer Service information Inquiry
    - Package: distribution
    - WSDL: CustomerServiceInquiry.wsdl
    - Implementation Class: CustomerServiceInquiryImpl.java

        - Public Operations / Methods: lookupCustomerService()
        - Implemented Via: JDBJ
    - Web Service: Sales Order Add / Update / Inquiry / Look up single SO
        - Package: distribution
        - WSDL: SalesOrder.wsdl
        - Implementation Class: SalesOrderImpl.java
        - Public Operations / Methods: getSalesOrder(), setSalesOrder(), lookupSalesOrderByCustomer()
        - Implemented Via: JDBJ
    - Web Service: Sales History information Inquiry
        - Package: distribution
        - WSDL: SalesHistoryInquiry.wsdl
        - Implementation Class: SalesHistoryInquiryImpl.java
        - Public Operations / Methods: lookupSalesHistory(),
        - Implemented Via: JDBJ
    - Web Service: Billing Instructions Line of Business Inquiry / Add / Update / Delete
        - Package: utilities
        - WSDL: LineOfBusiness.wsdl
        - Implementation Class: LineOfBusinessImpl.java
        - Public Operations / Methods: getCustomerLOB(), setCustomerLOB()
        - Implemented Via: Transidiom over screens V420631, V43063, V01053, V01153, V01054, V01154, V42063, V01054, V01154

Procurement

- Web Service: Purchase Order Inquiry / Add / Update / Look up single PO
    - Package: distribution
    - WSDL: PurchaseOrder.wsdl
    - Implementation Class: PurchaseOrderImpl.java
    - Public Operations / Methods: getPO(), setPO(), lookupPurchaseOrderBySupplier()
    - Implemented Via: JDBJ

Work Order

- Web Service: Work Order Inquiry
    - Package: utilities
    - WSDL: WorkOrder.wsdl

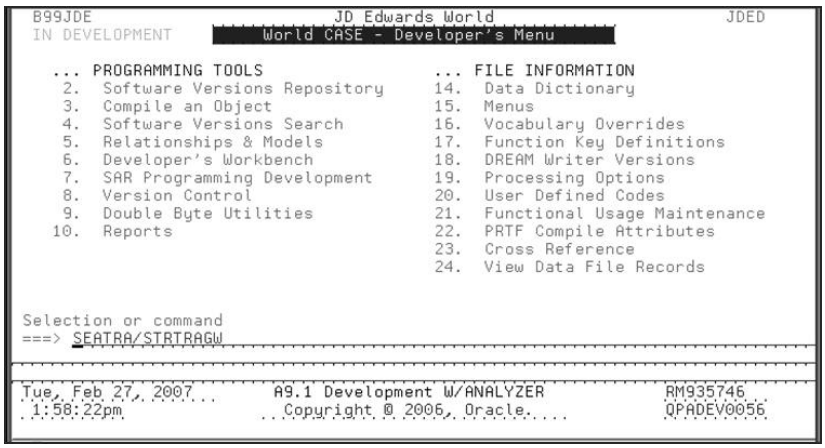
- Implementation Class: WorkOrderImpl.java
- Public Operations / Methods: getWorkOrder()
- Implemented Via: JDBJ

How to Turn Service Enablement on

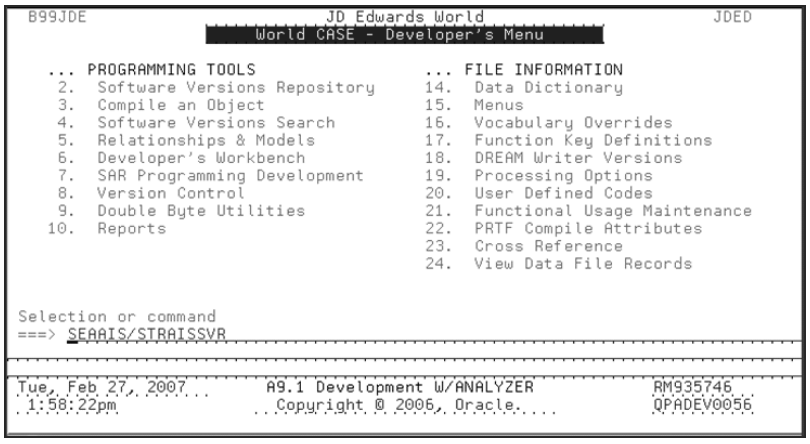
Starting the Transidiom Gateway and AIS

Once the Transidiom Gateway and AIS are installed, they can be started.

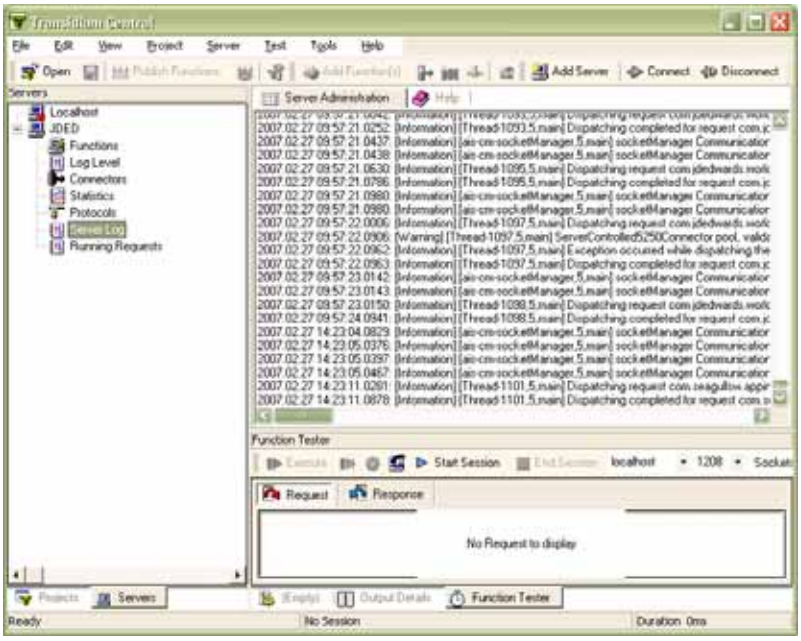
1. From the System i, the command to start the Transidiom Gateway is "SEATRA\STRTRAGW", which should be issued on a command line.



2. The command to start the AIS is "SEAAIS/STRAISSVR", which should also be issued on a command line.

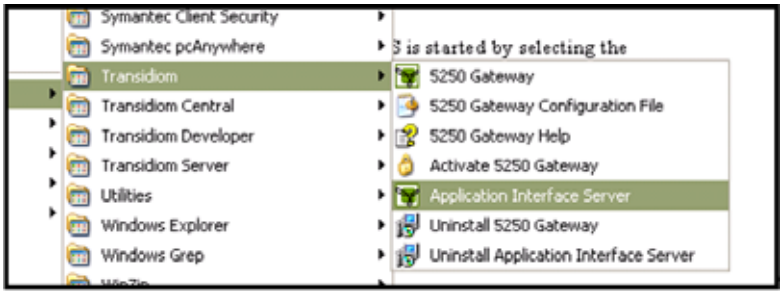


3. Once the Transidiom components are running, any problems can be discerned by evaluating the AIS log. This can be done either through Transidiom Central or by looking at the log file, which is located in the IFS directory on the System i at "SEAAIS\log".



4. From Windows, the Transidiom Gateway can be started by selecting the 5250 Gateway in the Transidiom folder in the Programs option of the Windows Start menu.

5. Starting the AIS in Windows is similar. The AIS is started by selecting the "application Interface Server" in the Transidiom folder in the Programs option of the Windows Start menu.



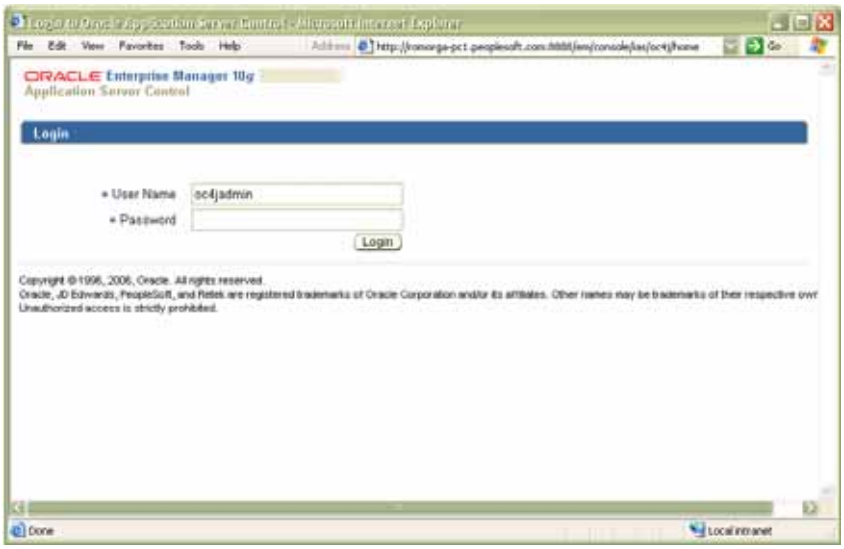
Deploying and Enabling Web Services

The following basic instructions are for deploying the JD Edwards World Service Enablement .ear application files to either the Oracle Application Server 10.1.3 (OAS) or the WebSphere Application Server 6.1 (WAS). You can also use WebSphere Application Server Express edition. See either the Oracle or WebSphere Application Server documentation for detailed instructions and deployment options on deploying the .ear files.

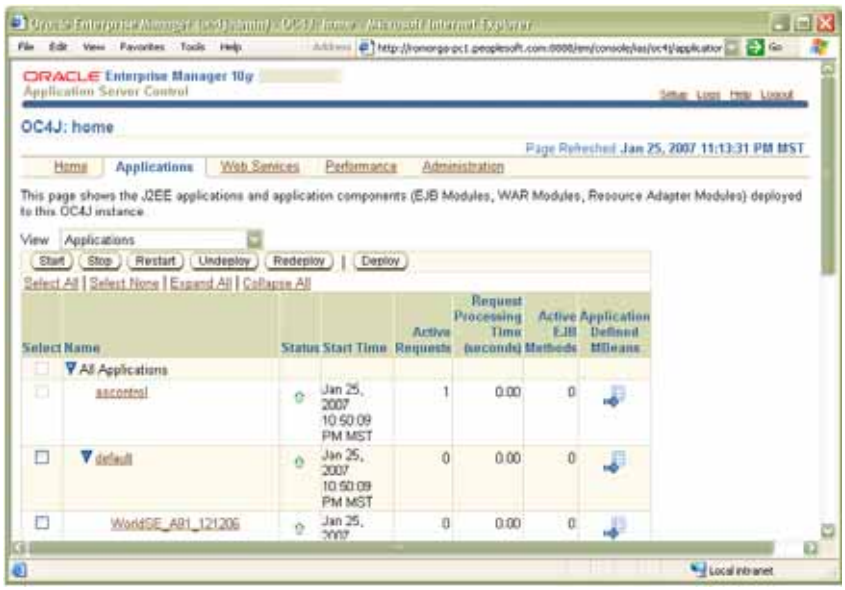
Deploying to Oracle Application Server

1. Start your Oracle Application Server.

2. Login to OAS Enterprise Manager. The following is the default: http://machine.name:8888/em



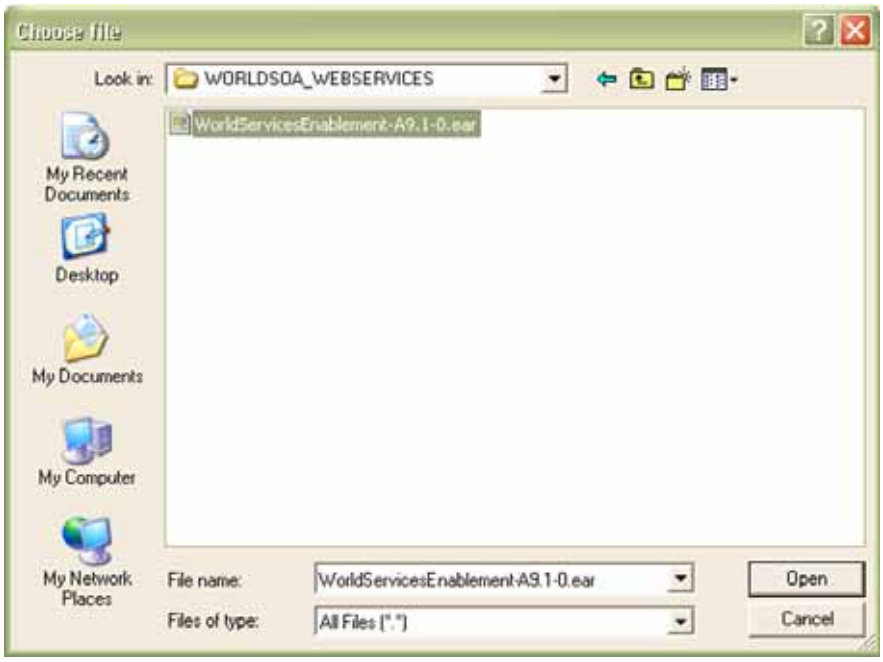
3. From the OC4J: home page, click Applications tab and then click Deploy.



4. From Deploy: Select Archive, click Deploy.

5. Click Browse to access the location for the Archive Location on the local host.

6. From Choose File, locate the WORLDSOA_WEBSERVICES folder in your install directory. Choose the WorldServicesEnablement-A9.1-0.ear file and click Open.

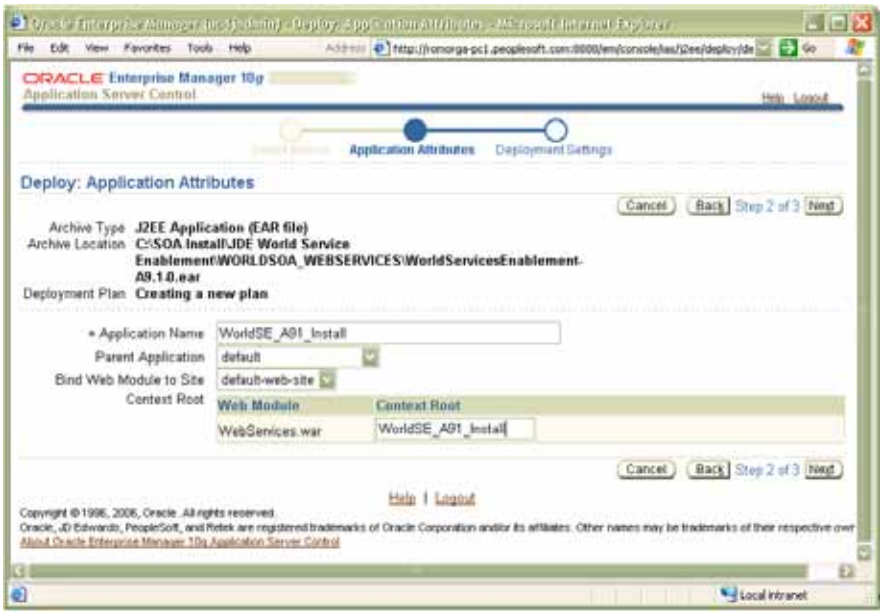


7. From Deploy: Select Archive, click Next.

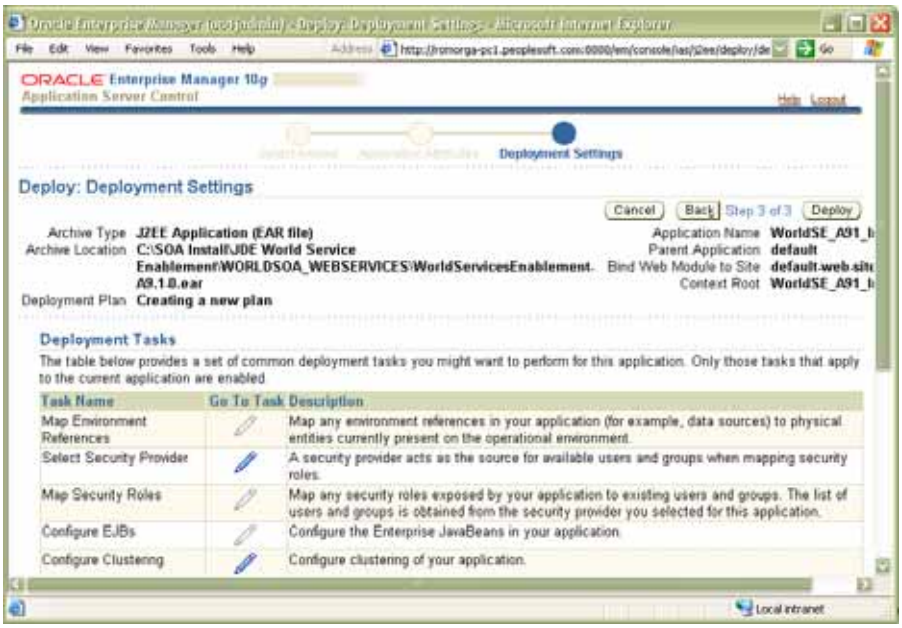
8. From Deploy: Application Attributes, complete the following field and click Next.

- Application Name

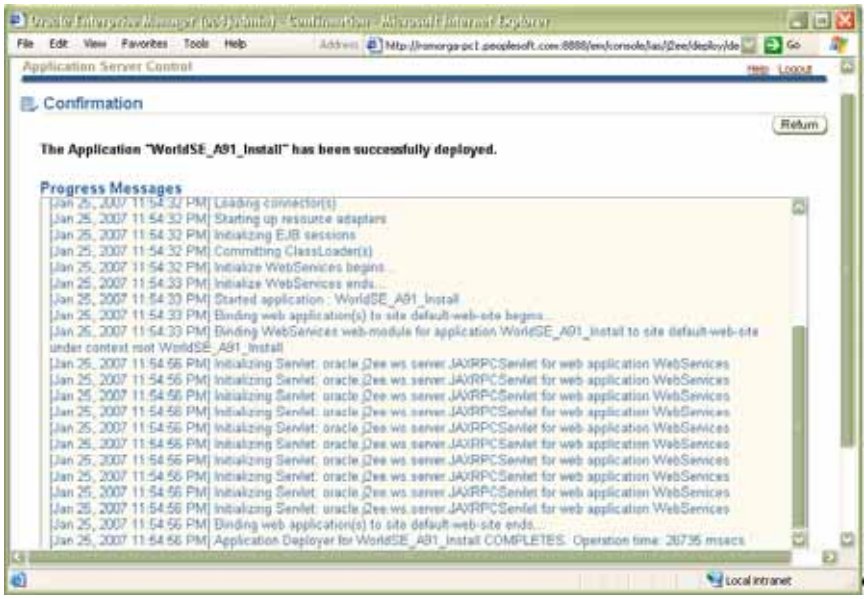
You can change the Context Root attribute (Optional).



9. From Deploy: Deployment Settings, click Deploy.



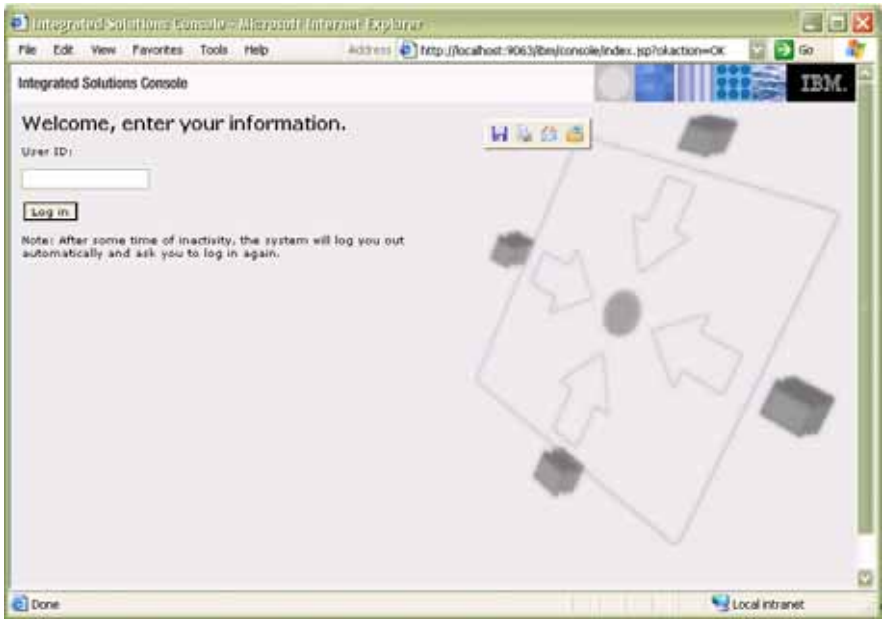
10. From the Confirmation screen, click Return.



Deploying to WebSphere Application Server

1. Start your WebSphere Application Server.

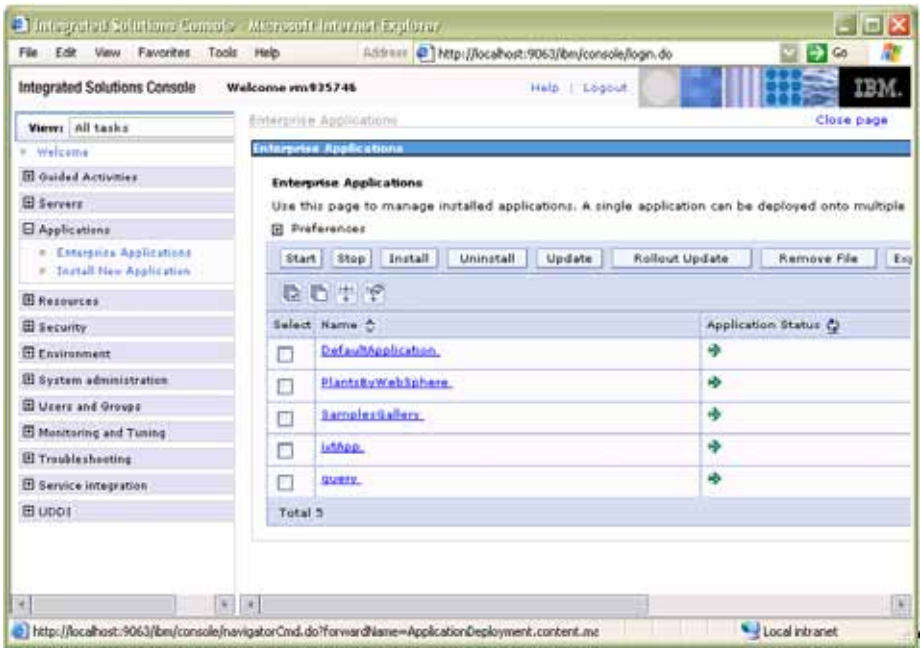
2. Login to WAS Administration Console. The following is the default: http://machine.name:port/ibm/console.



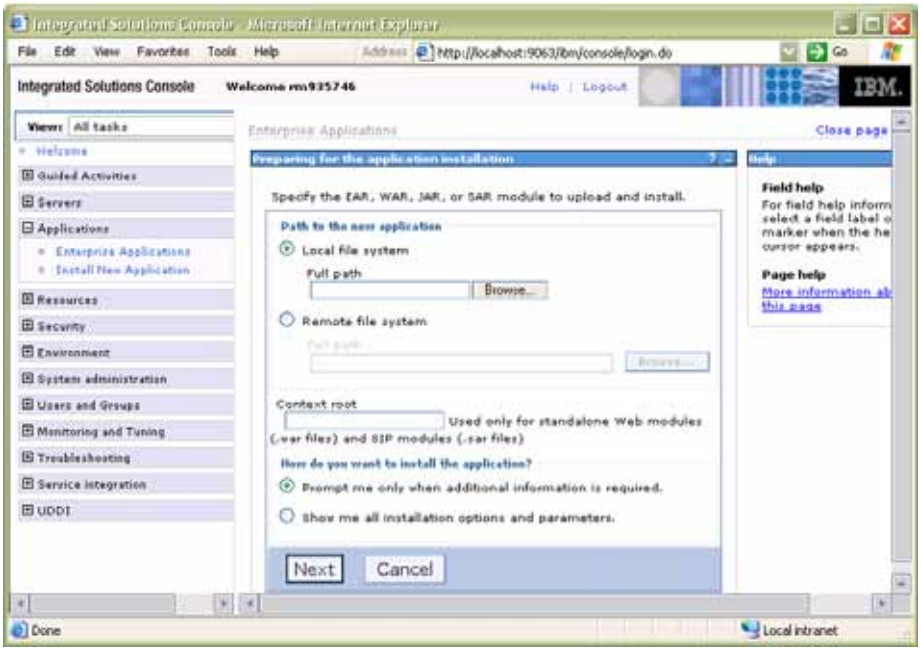
3. From Integrated Solutions Console Welcome, open the Applications option and choose Enterprise Applications.

   The system displays the Enterprise Applications.

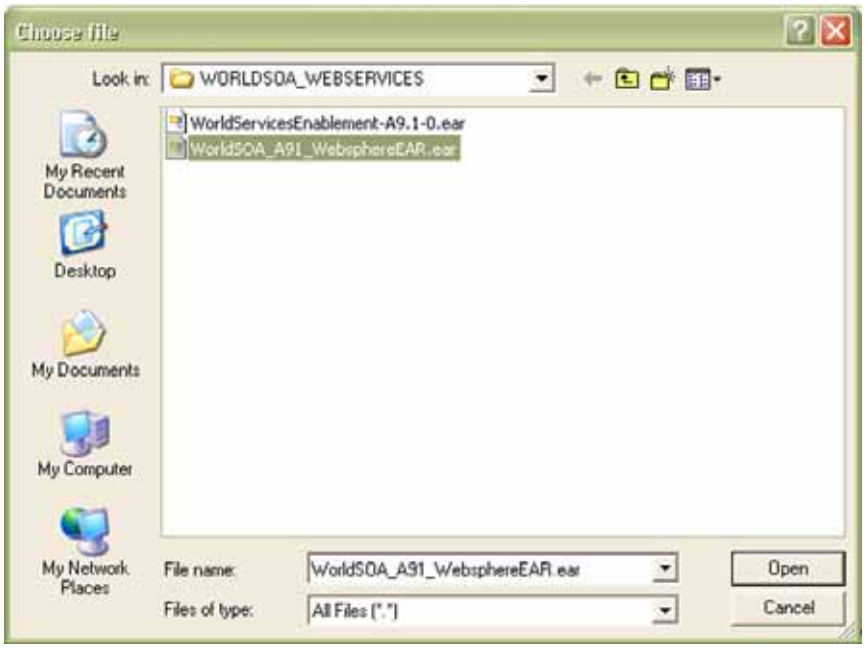
4. From Enterprise Applications, click the Install button.



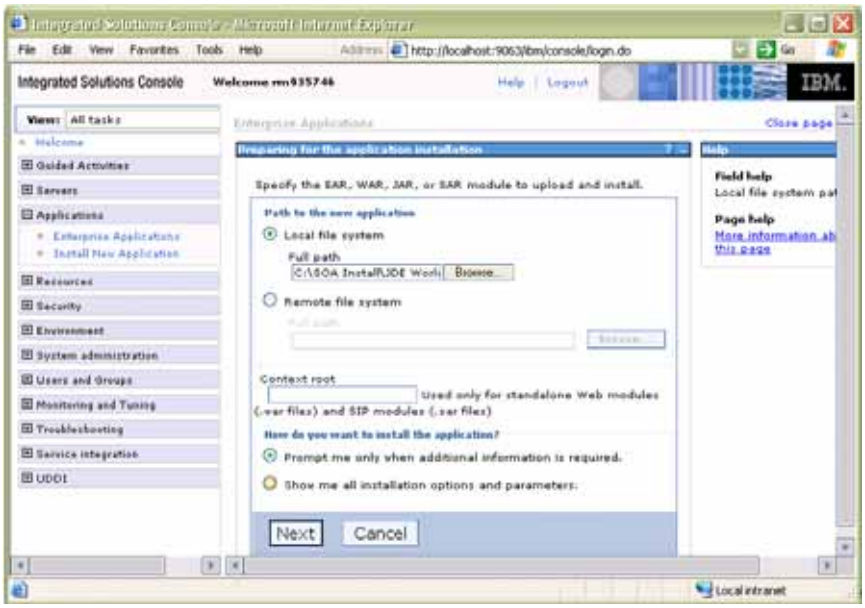
5. From Preparing for the application installation, choose Local file system and then click Browse to locate the file.



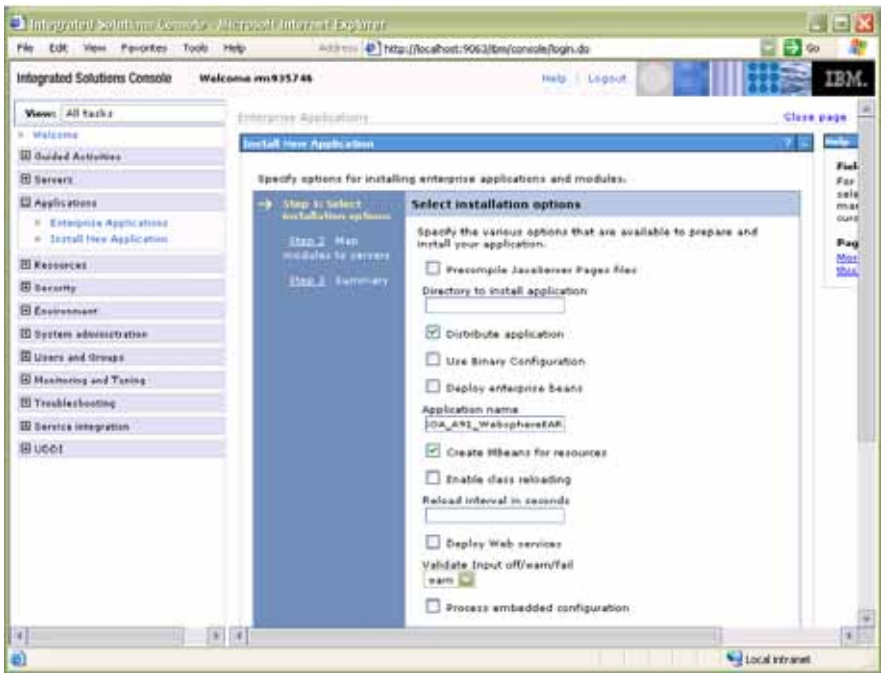
6. From Choose File, open the WORLDSOA_WEBSERVICES folder in your install directory, choose the WorldSOA_A91_WebsphereEAR.ear file and click Open.



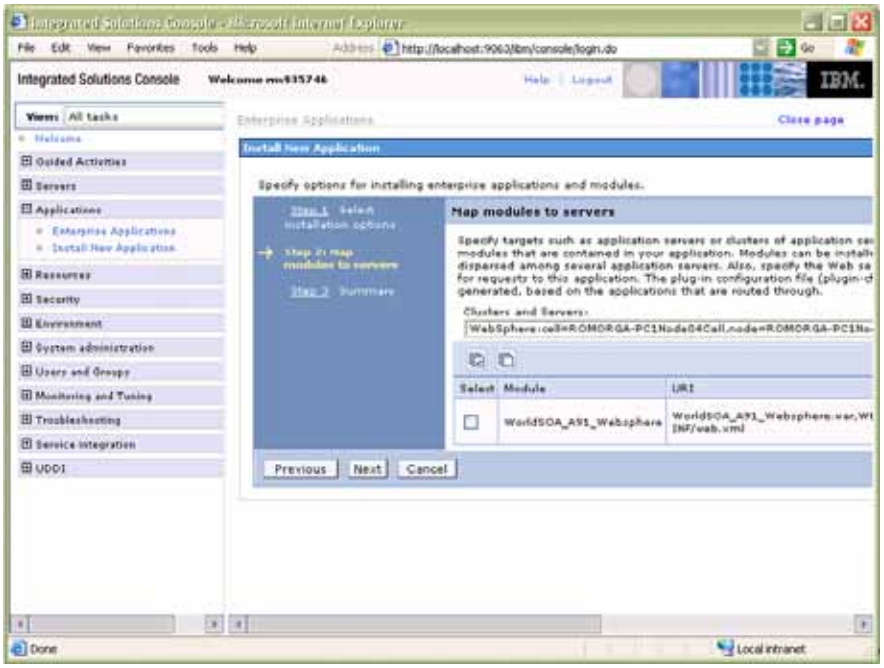
7. From Preparing for the application installation, click Next.



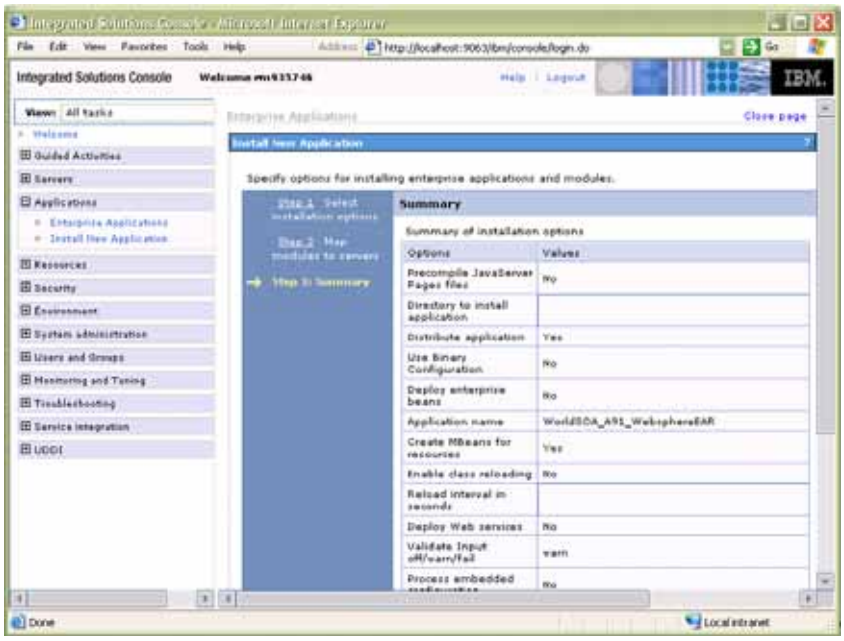
8. From Install New Application, accept the default values and click Next at the bottom of the screen.



9. From Install New Application, verify the correct server is set for the module, if multiple servers are running, and click Next.

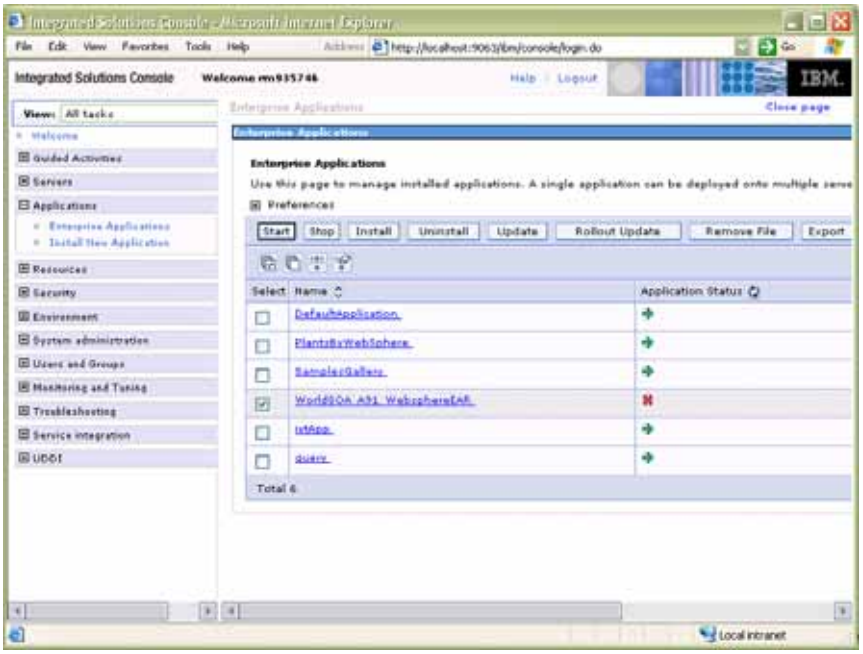


10. From Install New Application, click Finish at the bottom of the screen.

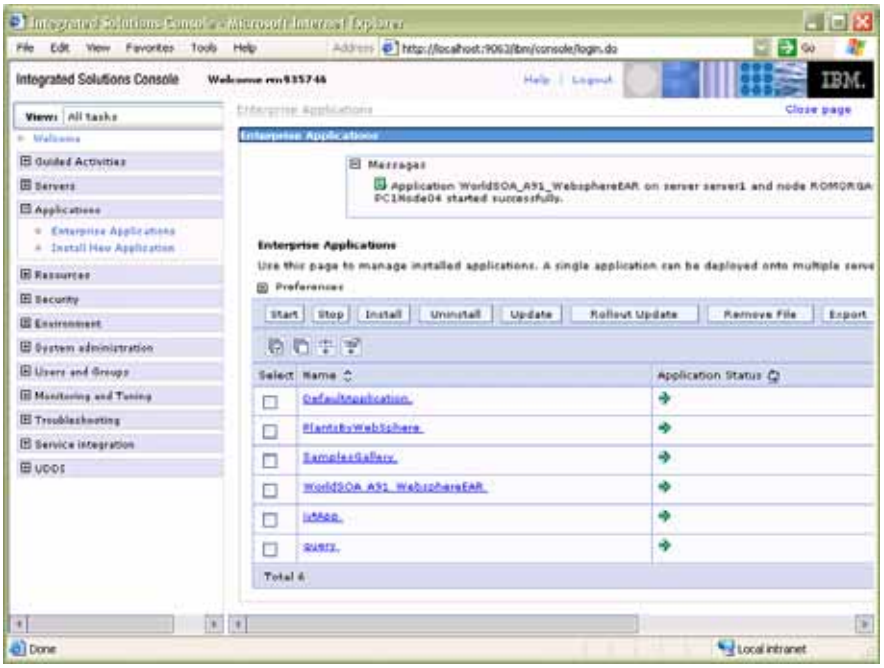


11. From Integrated Solutions Console Welcome, choose Save directly to the master configuration.

12. From Enterprise Applications, choose WorldSOA_A91_WebsphereEAR application and click Start.



13. From Enterprise Applications, the system displays the message that the application has started successfully.



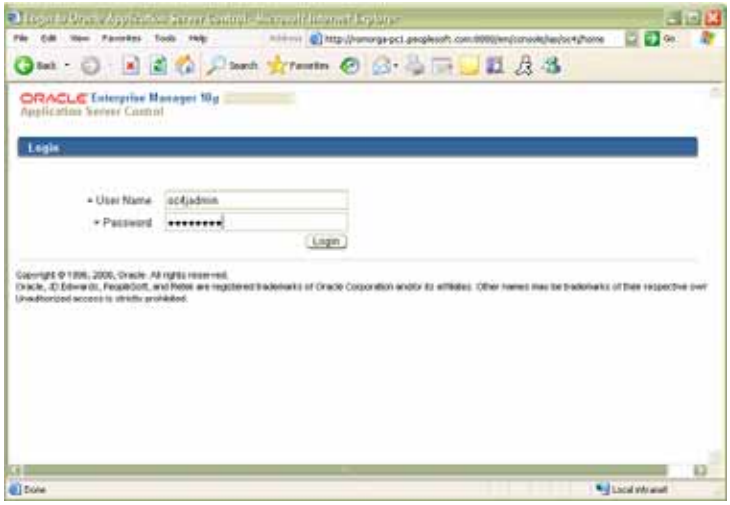
Testing Web Services

Testing the Web Services can take place in many ways. The next section details a testing scenario using the tools provided with the Oracle Application Server. Also

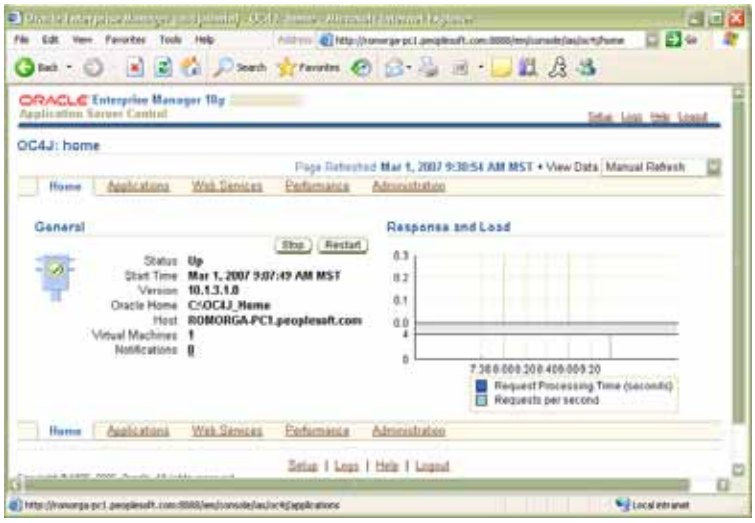
explained after that, is a detailed example of extracting the WSDL files for testing with the WebSphere Application. Another way to test the web services is to use an external application. For example, you could use an open source testing application like SoapUI.

Testing on Oracle Application Server

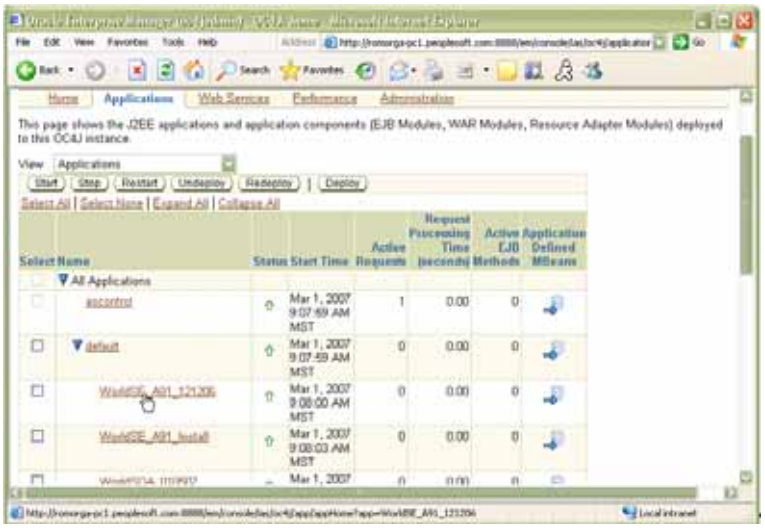
1. Login in to the Oracle Application Server (OAS) Enterprise Manager from the Login screen.



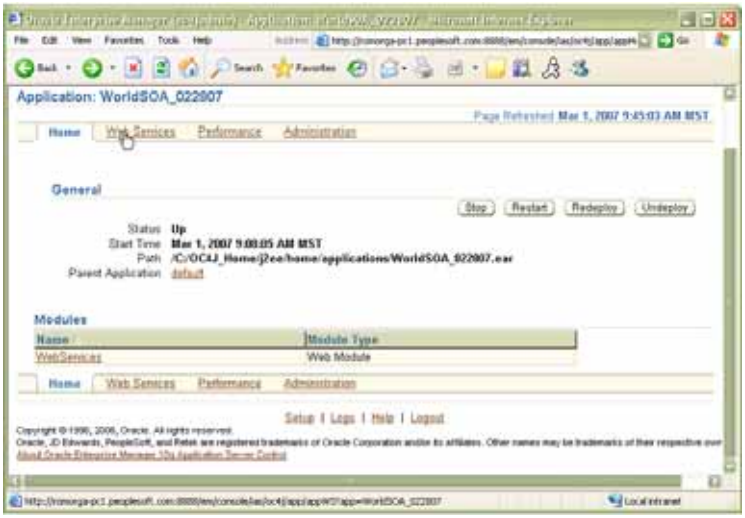
2. From the OC4J:home screen, select the Applications tab to see the installed applications.



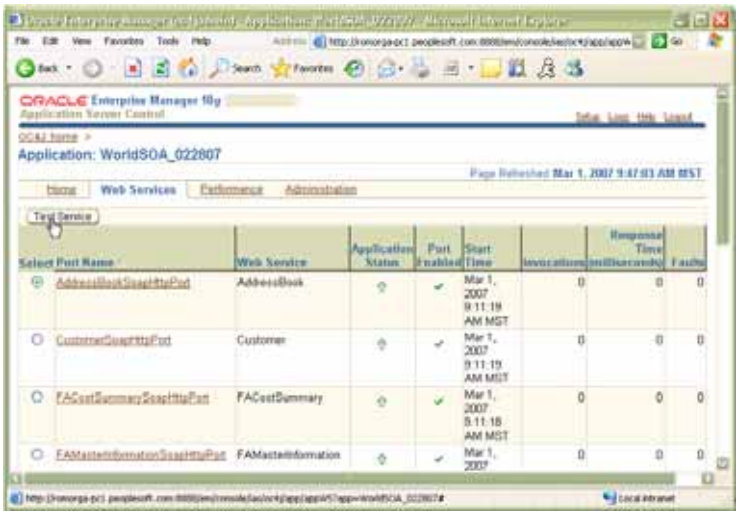
3. From the Applications tab screen, select the JD Edwards World web services application that is to be tested.



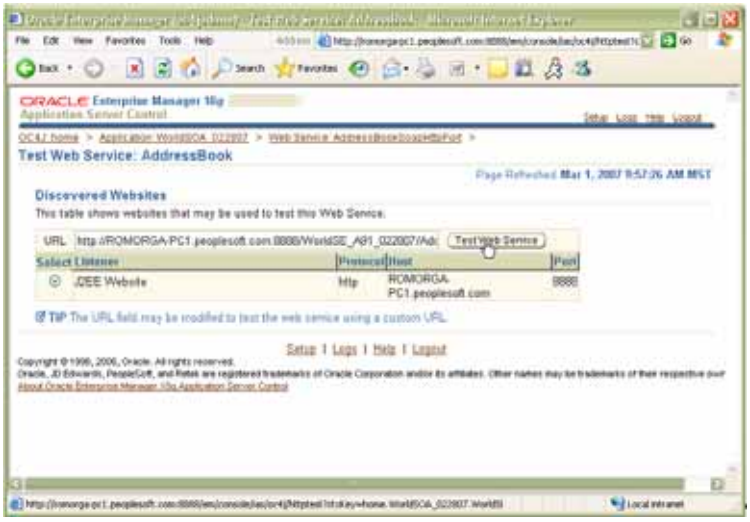
4. From the Application Home screen, select the Web Services tab.



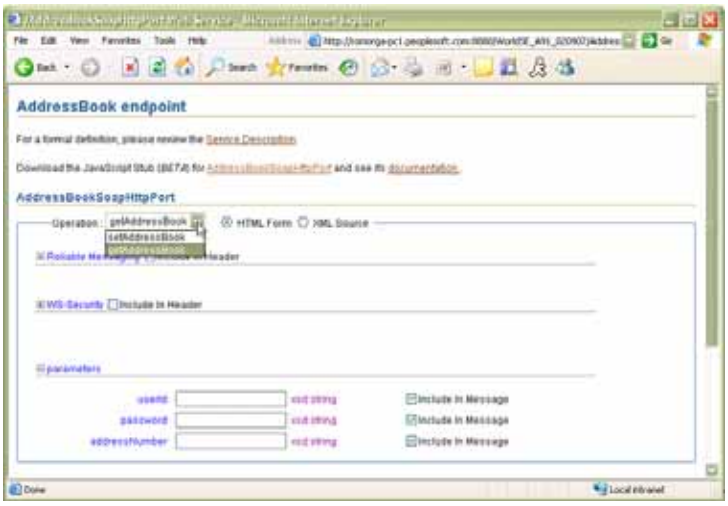
5. From the Application Web Service tab screen, select the radio button beside the web service you wish to test from the list provided. Then select the Test Service button at the top of the list.



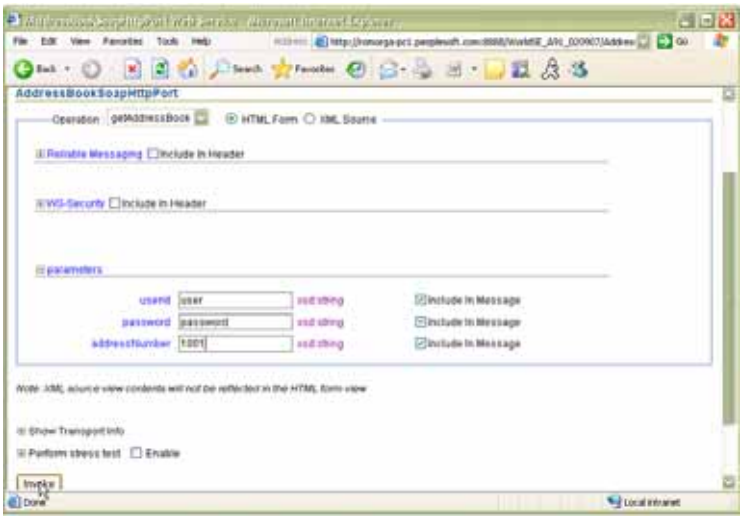
6. From the Discovered Websites screen, select the Test Web Service button. This should open a new browser window with the web service endpoint.



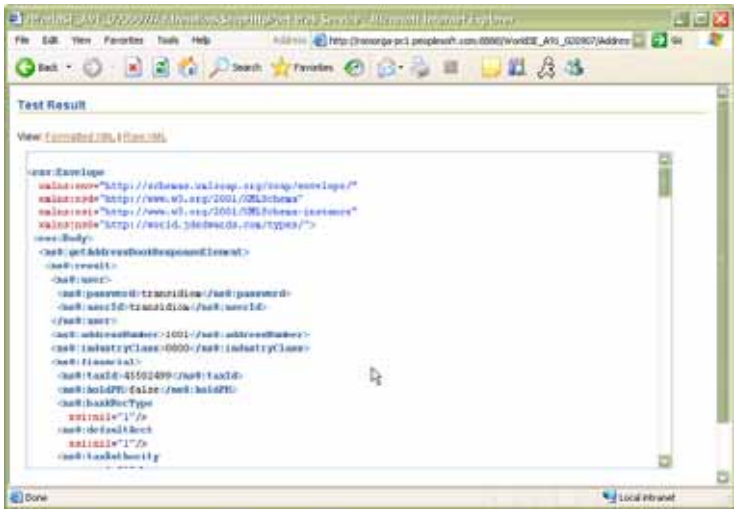
7. From the web service endpoint screen, select the operation you wish to test from the Operation drop down list. You can also select either HTML Form input or XML input.



8. Enter the parameters for the operation and the click Invoke at the bottom of the screen.

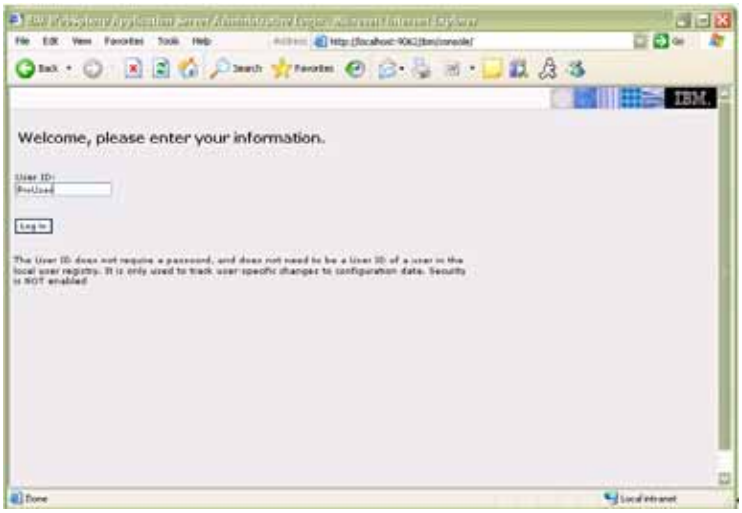


9. From the Test Result screen you will see the data returned from the web service in XML. You can select to have the XML formatted or in is raw form.

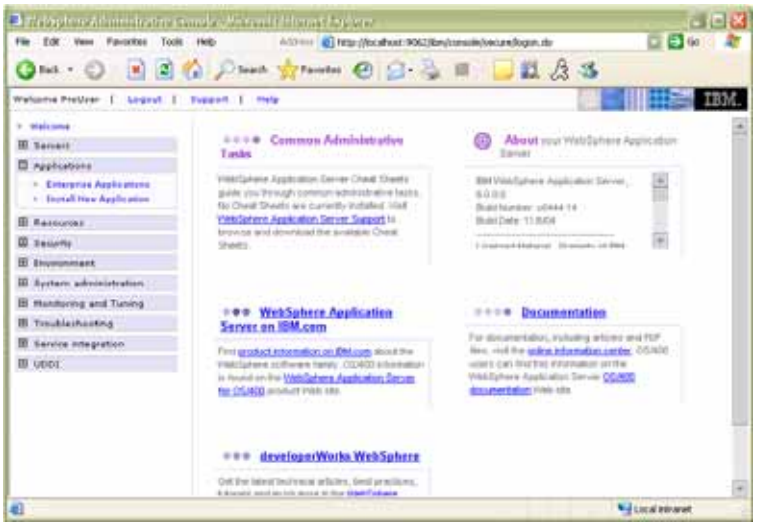


Extracting WSDL Files for Testing with WebSphere Application Server

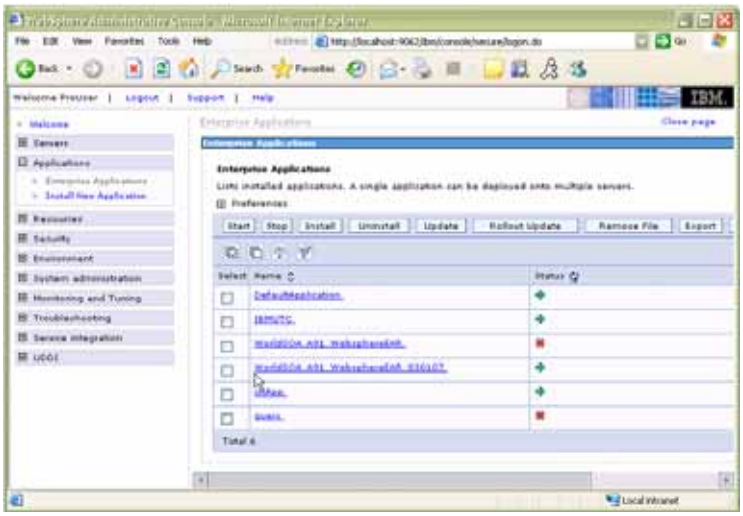
1. Login in to the WebSphere Application Server (WAS) Administrative Console from the Login screen.



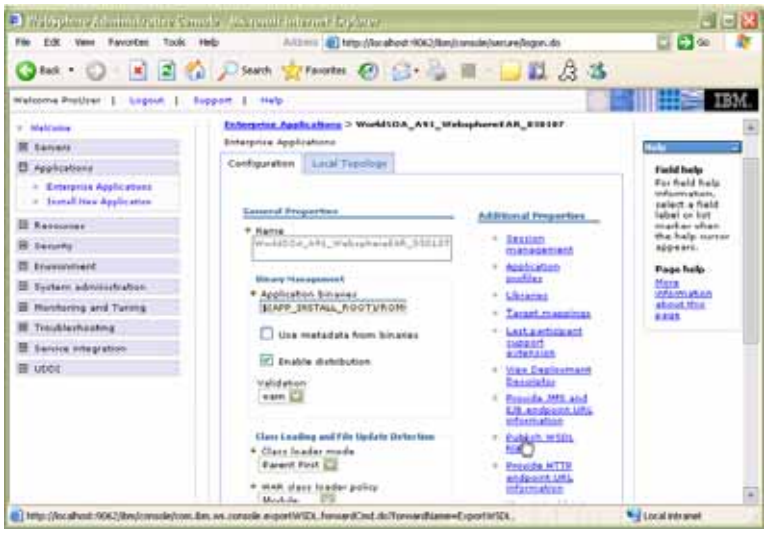
2. From the Welcome screen, expand Applications on the side bar. Select Enterprise Applications.



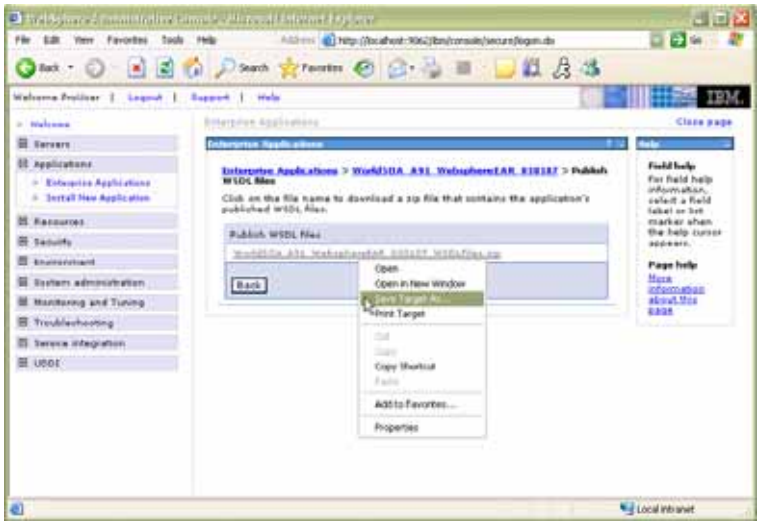
3. From the Enterprise Applications screen, select the application that you want to test.



4. From the Enterprise Applications properties screen, select the Publish WSDL files option on the right side of the screen.



5. From the Publish WSDL file screen, right-click the WSDL Files .zip link and select Save Target As. Save the file to a local directory.



6. The WSDL Files .zip file can be extracted on you local machine. The files can then be used for testing with an external application.

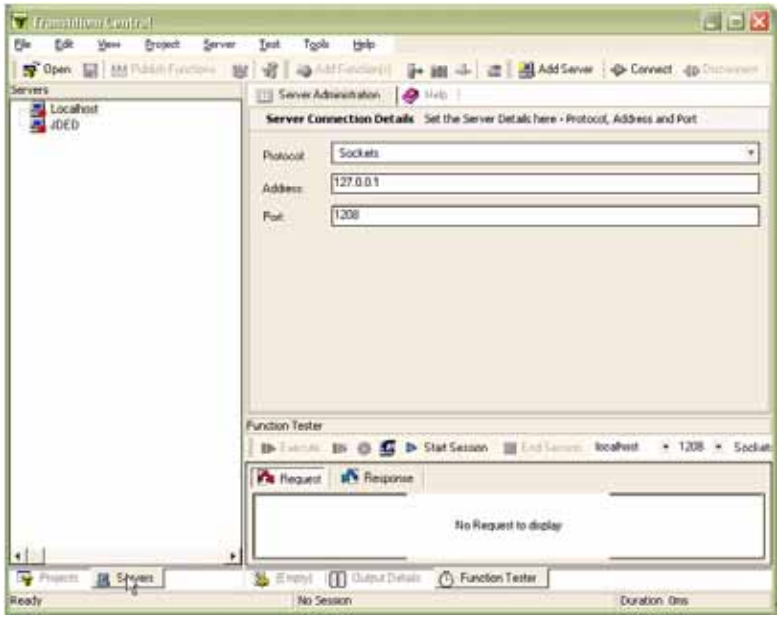
Components of a JD Edwards World Web Service

Transidiom Components

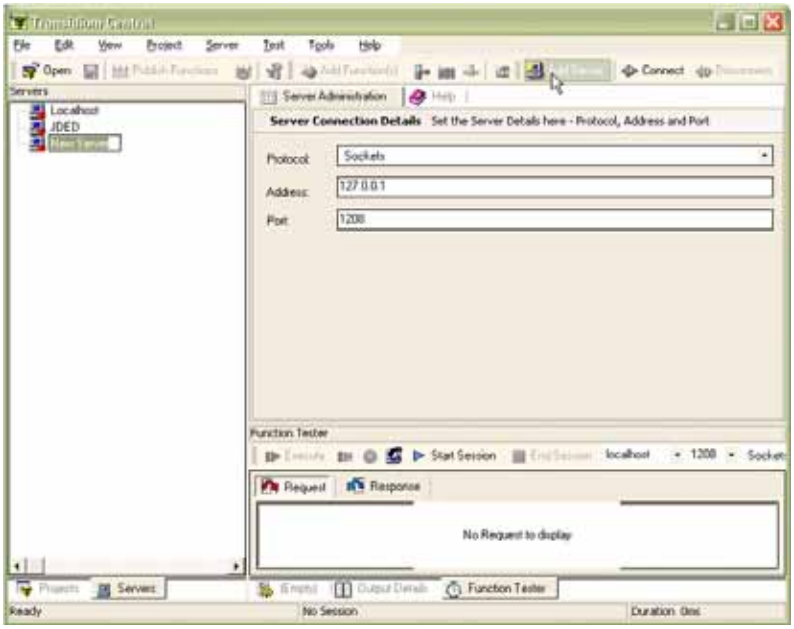
Transidiom allows JD Edwards World applications to run automatically with data that is scripted. The Transidiom scripts are developed using Seagull's 5250 Connector Builder product. The scripts are built starting from the sign on screen. They sign on to the environment, menu travel to the correct menu, select the application and enter the data supplied to the script. Once the operation is completed within an application, the return data is passed back out of the script. If an application has an error, it is trapped and the error information is passed back out of the script.

The services created for JD Edwards World may use one or many Transidiom scripts to complete the service. Most of the scripts created do a single task, like inquire on address book category codes or phone numbers. When a Web Service needs data from multiple applications the separate scripts are combined in the Java API layer to get the needed data.

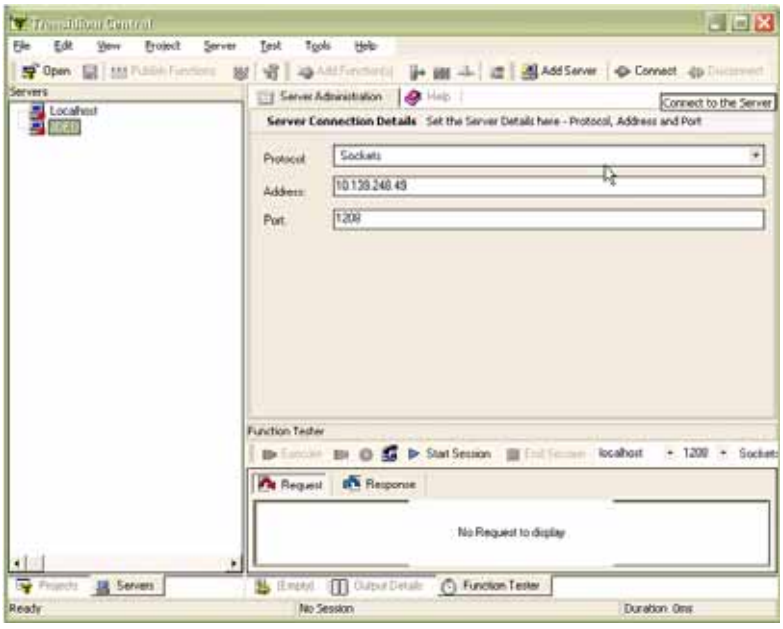
Once a script is created in the 5250 Connector Builder, it must be published to the Application Interface Server (AIS). When it is published to the AIS, then it can be tested using Transidiom Central. In Transidiom Central, connect to an AIS server that is running. You can do this by selecting the Servers Tab at the bottom of the left hand panel.



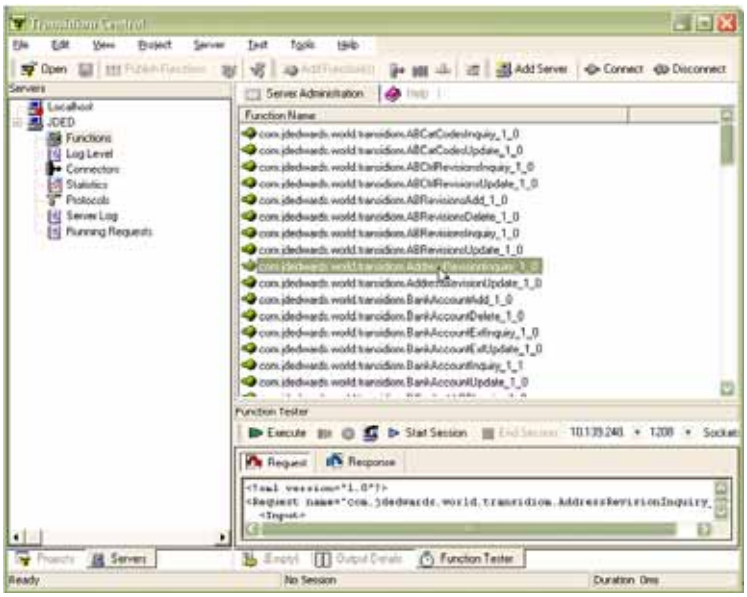
1. From the Servers tab, select the Add Server button from the toolbar at the top.



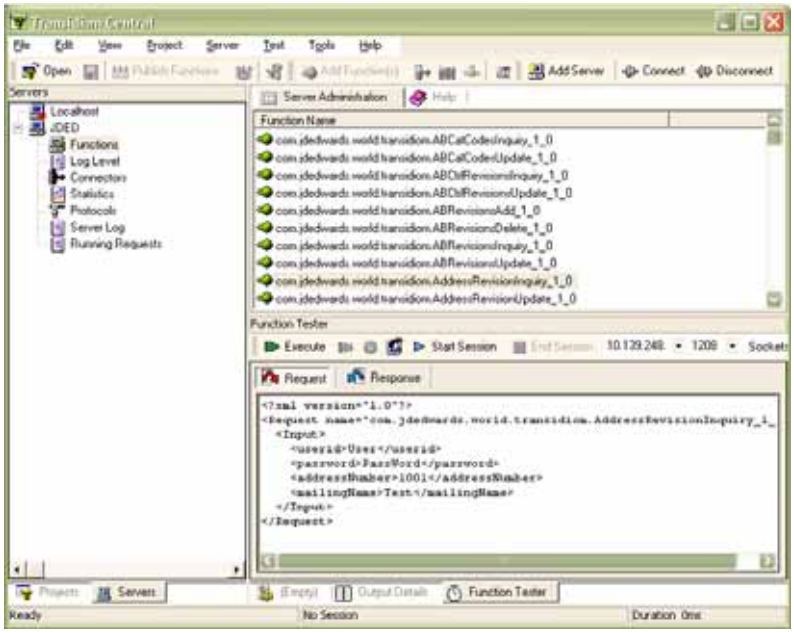
2. Enter the name of the new sever and enter the IP Address and Port number for the AIS you want to connect to. After entering the information, select the Connect button from the top toolbar to connect to the AIS.



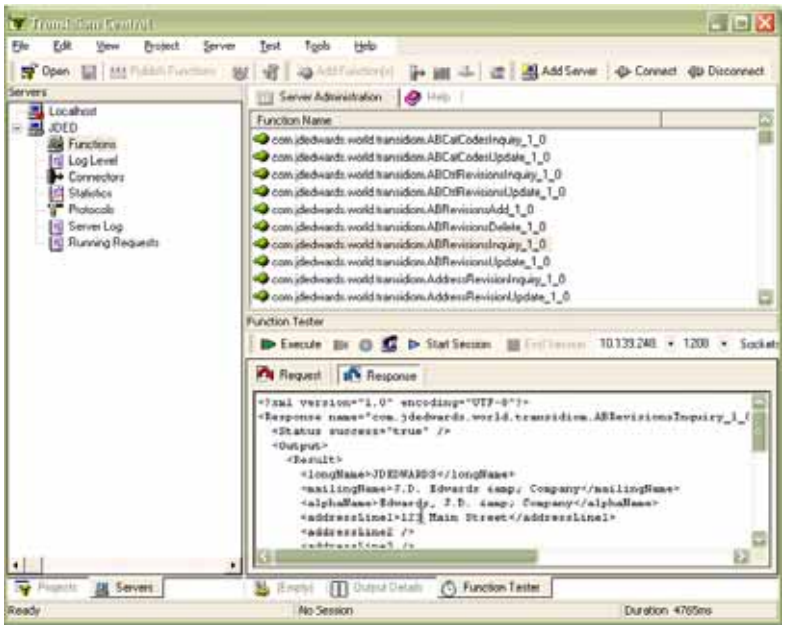
3. Once Transidiom Central is connected, you can select the Functions item from the left hand frame. Select the function (script) that you would like to test.



4. From the Function Tester panel of the screen, enter data between the XML tags and then select Execute to submit the data.



5. From the Request is processed the response data can be viewed in the Response frame.



6. When the scripts are published, Java command bean classes are also created. The command beans are imported into the Java API layer to allow for connectivity to the AIS.

Java Layer

The Java API layer used for JD Edwards World Service Enablement has several different parts including the above mentioned Transidiom Java command beans. Also included are the data objects that represent the data in JD Edwards World, implementation classes, connection classes and other supporting classes. The source code for the Java layer is included with the install. Both an Oracle JDeveloper project and an IBM WebSphere workspace have been provided.

When creating a Web Service, all of the above components need to be in place. The Transidiom command beans can be imported into your Java IDE project. The inputs and outputs from the Transidiom command beans can be used as a guide to create the data objects.

In some cases, instead of Transidiom, direct calls to JD Edwards World RPG programs and databases queries are used to collect the data. In theses cases, the IBM JT400 API and the JD Edwards World JDBj API is used to process the calls to the program and the database queries.

The implementation classes use the data objects as inputs and outputs. The logic within the implementation classes calls the Transidiom command beans (in some cases more than one) and combines the data returned from Transidiom to populate the output data objects.

The implementation classes are used to create the web services. The web services are created using the internal IDE tools. By doing this the WSDL and all the other necessary parts of the web service are created. Refer to your selected IDE documentation for further instructions on creating web services.

Retrofitting Web Services

In some cases, retrofitting the Web Services to a prior JD Edwards World release may be required. Though this would be a customization of the software, the following are some suggestions on how to approach the task.

Since the Transidiom scripts are built to coincide with the A9.1 release level, they will only work with that release. To retrofit to a prior release, new Transidiom scripts need to be created. If you duplicate current functionality with custom scripts for a prior release, study the current scripts for processing steps and data item names. Once the scripts are created, the command beans can be imported to the Java project.

It would be a good idea to create a copy of the Java project delivered so that when updates are issued changes are not overridden. Also, when creating classes to handle prior release processing, you should identify them differently from the base classes delivered.

Appendix A – Processing Options

Processing Options

Transidiom scripts are dependent upon the JD Edwards World application screens being consistent. Once the scripts are recorded if the screen display changes, a popup window is added to the screen or the process flow of the screen is changed, the Transidiom script will fail to work. In some applications, these screen aspects can be affected by processing options.

The following lists the applications and DREAM Writer versions used by Service Enablement. If the processing options are changed for any of these versions, it may cause the processing of the web services to fail in some way.

> **Caution:** Proceed with caution if you choose to change any of these versions in such a way that it affects the display and/or processing flow of the application.

Description	Form ID	Version No
Item Master Information - Revisions	P4101	ZJDES001
Address Book Information	P01051	ZJDES001
A/R Information	P01053	ZJDES001
Accounts Payable Information	P01054	ZJDES001
Address Revisions	P0116	ZJDES001
Asset Master Information	P1201	ZJDES001
Item Cost Summary	P122101	ZJDES001
Text Messages Review	P4016	ZJDES001
Text Messages Review	P4016	ZJDES002
Non-Stock Item Master Information - Revisions	P4101N	ZJDES001
Item Master Information - Manufacturing Data	P41013	ZJDES001
Item Branch Information - Revisions	P41026	ZJDES001
Item Branch Information - Manufacturing Data	P41027	ZJDES001
Item Cost Revisions	P4105	ZJDES001
Item Availability Summary	P41202	ZJDES001

Description	Form ID	Version No
Process Batch Journal Entries	P09110Z	XJDES001
Batch Order Edit and Creation - Sales	P40211Z	XJDES001
Batch Order Edit and Creation - Sales	P40211Z	XJDES002
EDI Purchase Order Change Edit/Create	P471311	XJDES001
EDI Purchase Order Change Edit/Create	P471311	XJDES002
Purchase Order Batch Entry Edit Object	P471411	XJDES001
Purchase Order Batch Entry Edit Object	P471411	XJDES002
Accounts by Business Unit	P0901	ZJDES001