# BEA WebLogic Adapter for Oracle Applications®

## User Guide

Version 8.1.0
Document Date: July 2003

# Copyright

## Restricted Rights Legend

## Trademarks or Service Marks

# Contents

## About This Document

Who Should Read This Documentation . . . xiii

Additional Information . . . xiv

How to Use This Document . . . xv

Contact Us! . . . xv

Documentation Conventions . . . xvii

## Introducing the BEA WebLogic Adapter for Oracle Applications

About Adapters and BEA WebLogic Integration . . . 1-2

Key Components of Integration Solutions . . . 1-3

- Basic WebLogic Integration Architecture . . . 1-3
- Enterprise Information Systems . . . 1-4
- Resource Adapters . . . 1-4
- Application Views . . . 1-5
- Service Clients and Event Consumers . . . 1-6
  - Service Clients . . . 1-6
  - Event Consumers . . . 1-7
- EIS Metadata, Schemas, and Repositories . . . 1-7
  - Schema . . . 1-7
  - Repositories . . . 1-9

About the BEA WebLogic Adapter for Oracle Applications . . . 1-9

- Oracle Applications Enterprise Application Integration Architecture . . . 1-9

Supported Oracle Applications Operations for Application Integration . . . . . . . . . 1-10
Benefits . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1-11
Getting Started With Adapter for Oracle Applications. . . . . . . . . . . . . . . . . . . . . . . . . . 1-11
Step 1: Design the Application Integration Solution . . . . . . . . . . . . . . . . . . . . . . . 1-12
Step 2: Determine the Required Oracle Applications Business Processes . . . . . . . 1-12
Step 3: Generate Schemas for Oracle Applications Integration Objects . . . . . . . . . 1-13
Step 4: Define Application Views and Configure Services and Events. . . . . . . . . . 1-13
Step 5: Integrate with Other BEA Software Components . . . . . . . . . . . . . . . . . . . . 1-14
Step 6: Deploy the Solution to the Production Environment. . . . . . . . . . . . . . . . . . 1-14

## Generating Schemas for Oracle Applications Integration

Before You Begin . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2-2
About the BEA Application Explorer. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2-2
About the Process for Defining Schemas . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2-2
Types of Schemas Generated by the BEA Application Explorer . . . . . . . . . . . . . . . 2-3
Service Requests. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2-3
Service Responses. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2-3
Events . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2-3
Concurrent Programs . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2-3
Starting the BEA Application Explorer . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2-4
Setting the Session Path . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2-4
Managing Oracle Applications Connections. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2-5
About Oracle Applications Logon Parameters . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2-5
Creating a New Connection . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2-6
Using an Existing Connection . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2-7
Disconnecting from Oracle Applications. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2-7
Removing Connections . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2-8
Managing Schemas . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2-8

Creating Schemas for Services . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2-8

Creating Schemas for Events . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2-10

Creating Schemas for the Concurrent Programs Agent . . . . . . . . . . . . . . . . . . . . . 2-13

Removing Schemas . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2-13

Next Steps . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2-14

Defining Application Views for Oracle Applications

How to Use This Document . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3-2

Before You Begin . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3-2

About Application Views . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3-3

About Defining Application Views . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3-3

Defining Service Connection Parameters . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3-5

Setting Service Properties . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3-7

Interface Table . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3-7

AQService . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3-8

Concurrent Programs Agent . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3-10

Common Service and Event Settings . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3-12

Setting Event Properties . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3-13

OracleAppsTables . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3-13

Using Formats . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3-15

Using SQL Post Queries . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3-16

Using Delete Keys . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3-17

OracleAQ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3-17

Defining Event Connection Parameters . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3-19

Testing Services . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3-21

Testing Events Using a Service . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3-22

Testing Events Manually . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3-23

Testing the Concurrent Program Agent . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3-23

Determining the Request Parameters . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3-24

Creating the Request Document . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3-27

Submitting the Request and Viewing the Response. . . . . . . . . . . . . . . . . . . . . . . . . 3-30

## Processing Oracle Events

Listening for Newly Inserted Rows Using Row Tracking Events . . . . . . . . . . . . . . . . . . 4-1

About the Source Table . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4-1

About the Control Table. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4-2

Configuring an Event for Row Tracking . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4-2

Listening for Newly Inserted Rows Using Row Removal Events . . . . . . . . . . . . . . . . . 4-2

Configuring an Event for Row Removal . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4-3

Listening for Table Changes Using Trigger-Based Events. . . . . . . . . . . . . . . . . . . . . . . . 4-3

About Triggers . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4-3

Sample Trigger . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4-4

About the Control Table. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4-5

Configuring an Event for Trigger-Based Processing . . . . . . . . . . . . . . . . . . . . . . . . 4-5

## Oracle Interface Tables

Oracle Financials. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .A-1

Oracle Manufacturing/Distribution . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .A-3

## XML Business Functionality in the BEA WebLogic Adapter for Oracle Applications

BEA WebLogic Adapter for Oracle Applications and Oracle Functionality . . . . . . . . . . B-1

XML Business Functionality Descriptions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-2

Post Journal . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-2

Confirm . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-3

Process PO . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-3

Acknowledge PO . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-4

Load Receivable . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-4
Load Payable . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-5
Sync Customer . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-5
Sync Supplier . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-5
Load LdgrBudget . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-6
Get PO and Show PO . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-6
Receive PO . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-6
Get Prodorder . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-7
Show Prodorder and Receive Prodorder . . . . . . . . . . . . . . . . . . . . B-7
Transfer Item . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-7
Get Issueinfo and Show_IssueInfo . . . . . . . . . . . . . . . . . . . . . . . . B-8
Confirm Issue . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-8
Issue MiscItem and Receive MiscItem. . . . . . . . . . . . . . . . . . . . . . B-9
Get Countinfo and Show Countinfo . . . . . . . . . . . . . . . . . . . . . . . B-9
Update Invencount . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-9
Get Credit, Show Credit, and Update_credit . . . . . . . . . . . . . . . . . B-9
Change Status . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-10
Load Projacctg . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-10
Sync Projinfo . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-10
Get Wipconfirm and Show Wipconfirm. . . . . . . . . . . . . . . . . . . . . B-11
Update Wipconfirm . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-11
Get PickList . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-11
List PickList . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-12
Show PickList. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-12
Update PickList . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-13
Get Personnel and Show Personnel . . . . . . . . . . . . . . . . . . . . . . . B-13
Update Persontime . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-13
Sync Field. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-13

Sync Personnel . . . . . . . . . . B-13

Sync Wrkschdule . . . . . . . . . . B-14

Sync Mfgtlcode . . . . . . . . . . B-14

Sync COA . . . . . . . . . . B-14

Sync Exchngrate . . . . . . . . . . B-15

Add Requisitn . . . . . . . . . . B-15

Change Requisitn . . . . . . . . . . B-15

Cancel Requisitn . . . . . . . . . . B-16

Get Requisitn and Show Requisitn . . . . . . . . . . B-16

Getlist Requisitn . . . . . . . . . . B-16

List Requisitn . . . . . . . . . . B-17

Getlist PO . . . . . . . . . . B-17

List PO . . . . . . . . . . B-18

Add PO . . . . . . . . . . B-18

Change PO . . . . . . . . . . B-19

Cancel PO . . . . . . . . . . B-19

Update Delivery . . . . . . . . . . B-19

Update Inspection . . . . . . . . . . B-20

Sync Item . . . . . . . . . . B-20

Sync Sitelevel . . . . . . . . . . B-20

Get Prodavil . . . . . . . . . . B-21

Show Prodavil . . . . . . . . . . B-21

Update ProductReq . . . . . . . . . . B-22

Create Prodorder . . . . . . . . . . B-22

Cancel Prodreq . . . . . . . . . . B-23

Sync Inventory . . . . . . . . . . B-23

Get BOM and Show BOM . . . . . . . . . . B-23

Sync BOM . . . . . . . . . . B-24

Allocate Activity . . . . . . . . . . B-24
Load Matchdoc. . . . . . . . . . . B-24
Update Matchok and Update Matchfail . . . . . . . . . . B-25
Load_PLInvoice and Load Payable . . . . . . . . . . B-26
Get Matchdoc and Show Matchdoc . . . . . . . . . . B-26
Getlist Picklist . . . . . . . . . . B-26
Update Inventory . . . . . . . . . . B-27
Getlist Countinfo and List Countinfo. . . . . . . . . . B-27
Cancel Prodorder . . . . . . . . . . B-27
Sync Salesorder . . . . . . . . . . B-28
Add Salesorder . . . . . . . . . . B-28
Cancel Salesorder. . . . . . . . . . B-28
Change Salesorder . . . . . . . . . . B-29
Get Salesorder and Show Salesorder . . . . . . . . . . B-29
Getlist Salesorder and List Salesorder . . . . . . . . . . B-29
Sync PO . . . . . . . . . . B-30
Sync Routing . . . . . . . . . . B-30
Get Routing and Show Routing . . . . . . . . . . B-31
Getlist Routing and List Routing . . . . . . . . . . B-31
Getlist BOM and List BOM . . . . . . . . . . B-31
Get Item and Show Item. . . . . . . . . . B-32
Getlist Item and List Item. . . . . . . . . . B-32
Getlist Prodorder and List Prodorder . . . . . . . . . . B-33
Sync Prodorder. . . . . . . . . . B-33
Sync Engchgordr . . . . . . . . . . B-34
Get Engchgordr and Show Engchordr . . . . . . . . . . B-34
Confirm Engchordr. . . . . . . . . . B-35
Sync Dspchlist . . . . . . . . . . B-35

Get Dspchlist and Show Dspchlist . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-35
Update Dspchlist . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-36
Sync Maintorder . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-36
Create Maintorder . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-36
Update Maintorder . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-37
Cancel Maintorder . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-37
Get Maintorder and Show Maintorder . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-37
Getlist Maintorder List Maintorder . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-38
Sync UOMGroup . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-38
Get UOMGroup and Show UOMGroup . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-39
Getlist UOMGroup and List UOMGroup . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-39
Sync Catalog . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-40
Get Catalog and Show Catalog . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-40
Sync Itemclass . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-41
Get Itemclass and Show Itemclass . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-41
Sync ITEMXREF . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-42
Get ITEMXREF and Show ITEMXREF . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-43
Sync Pricelist . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-44
Get Pricelist and Show Pricelist . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-45
Sync Itemspecs . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-46
Get Itemspecs and Show Itemspecs . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-46
Sync RFQ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-47
Add RFQ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-48
Change RFQ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-48
Cancel RFQ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-48
Respond RFQ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-49
Getlist RFQ and List RFQ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-49
Get RFQ and Show RFQ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . B-50

Sync Quote . . . . . . . . . . B-51
Add Quote . . . . . . . . . . B-51
Change Quote . . . . . . . . . . B-51
Cancel Quote . . . . . . . . . . B-52
Respond Quote . . . . . . . . . . B-52
Getlist Quote and List Quote . . . . . . . . . . B-53
Get Quote and Show Quote . . . . . . . . . . B-53
Show Shipment. . . . . . . . . . . B-54
Sync Planschd. . . . . . . . . . . B-54
Get Planschd and Show Planschd. . . . . . . . . . . B-55
Sync Seqschd (Sequence Schedule) . . . . . . . . . . B-56
Get Seqschd (Sequence Schedule) and
Show Seqschd (Sequence Schedule). . . . . . . . . . . B-56
Sync Shipschd (Shipment Schedule) . . . . . . . . . . B-57
Get Shipschd (Shipment Schedule) and
Show Shipschd (Shipment Schedule) . . . . . . . . . . B-57
Process Invoice. . . . . . . . . . . B-57
Process WIPSPLIT. . . . . . . . . . . B-58
Process WIPMERGE . . . . . . . . . . B-58
Process WIPRECOVER. . . . . . . . . . . B-59
Get WIPSTATUS and Show WIPSTATUS . . . . . . . . . . B-59
Process WIPMOVE . . . . . . . . . . B-60
Allocate Resource. . . . . . . . . . . B-61
Get Customer and Show Customer . . . . . . . . . . B-61
Get Supplier and Show Supplier . . . . . . . . . . B-61
Sync Ecatalog. . . . . . . . . . . B-62
Get ECATALOG and Show ECATALOG . . . . . . . . . . B-62
Sync Invoice. . . . . . . . . . . B-63

Get Invoice and Show Invoice . . . . . . . . . . B-63
Get Consumption and Show Consumption . . . . . . . . . . B-64
Create Requisitn . . . . . . . . . . B-64
Getlist LDGRACTUAL and List LDGRACTUAL . . . . . . . . . . B-65
Get LDGRACTUAL and Show LDGRACTUAL . . . . . . . . . . B-65
Acknowledge Delivery . . . . . . . . . . B-66
Receive Delivery . . . . . . . . . . B-66
Get Delivery and Show Delivery . . . . . . . . . . B-67
Getlist Delivery and List Delivery . . . . . . . . . . B-68

## Samples

About the Sample Schemas . . . . . . . . . . C-1
About the Directory Structure . . . . . . . . . . C-1
About the Manifest File . . . . . . . . . . C-2
About the Request Schema . . . . . . . . . . C-3
About Sample XML to Test a Service . . . . . . . . . . C-3

## Index

# About This Document

This document describes how to install, configure, and use the BEA WebLogic Adapter for Oracle Applications. This document is organized as follows:

- Chapter 1, "Introducing the BEA WebLogic Adapter for Oracle Applications," describes the adapter and how it relates to both Oracle Applications and WebLogic Integration.
- Chapter 2, "Generating Schemas for Oracle Applications Integration," describes how to generate schemas for your Oracle Applications business objects using the BEA Application Explorer.
- Chapter 3, "Defining Application Views for Oracle Applications," describes application views and how to use them to configure events and services.
- Chapter 4, "Processing Oracle Events," describes how to use the adapter to process Oracle Applications events.
- Appendix A, "Oracle Interface Tables," lists the Oracle interface tables you can use.
- Appendix B, "XML Business Functionality in the BEA WebLogic Adapter for Oracle Applications," describes the Oracle business documents that are available to the adapter.

## Who Should Read This Documentation

This document is intended for the following members of an integration team:

- Integration Specialists—Lead the integration design effort. Integration specialists have expertise in defining the business and technical requirements of integration projects, and in designing integration solutions that implement specific features of WebLogic Integration.

The skills of integration specialists include business and technical analysis, architecture design, project management, and WebLogic Integration product knowledge.

- Technical Analysts—Provide expertise in an organization's information technology infrastructure, including telecommunications, operating systems, applications, data repositories, future technologies, and IT organizations. The skills of technical analysts include technical analysis, application design, and information systems knowledge.
- Enterprise Information System (EIS) Specialists—Provide domain expertise in the systems that are being integrated using WebLogic adapters. The skills of EIS specialists include technical analysis and application integration design.
- System Administrators—Provide in-depth technical and operational knowledge about databases and applications deployed in an organization. The skills of system administrators include capacity and load analysis, performance analysis and tuning, deployment topologies, and support planning.

## Additional Information

To learn more about the software components associated with the adapter, see the following documents:

- *BEA WebLogic Adapter for Oracle Applications Release Notes*

  `http://edocs.bea.com/wladapters/docs81/pdf/relnotes.pdf`
- *BEA WebLogic Adapter for Oracle Applications Installation and Configuration Guide*

  `http://edocs.bea.com/wladapters/docs81/pdf/install.pdf`
- *BEA Application Explorer Installation and Configuration Guide*

  `http://e-docs.bea.com/wladapters/bae/docs81/pdf/install.pdf`
- *Introduction to the BEA WebLogic Adapters*

  `http://e-docs.bea.com/wladapters/docs81/pdf/intro.pdf`
- BEA WebLogic Adapters 8.1 Dev2Dev Product Documentation

  `http://dev2dev.bea.com/products/product.jsp?highlight=wla`
- Application Integration documentation

  `http://edocs.bea.com/wli/docs81/aiover/index.html`

  `http://edocs.bea.com/wli/docs81/aiuser/index.html`
- BEA WebLogic Integration documentation

http://edocs.bea.com/wli/docs81/index.html

- BEA WebLogic Platform documentation

  http://edocs.bea.com/platform/docs81/index.html

- Oracle Applications documentation

  www.oracle.com

## How to Use This Document

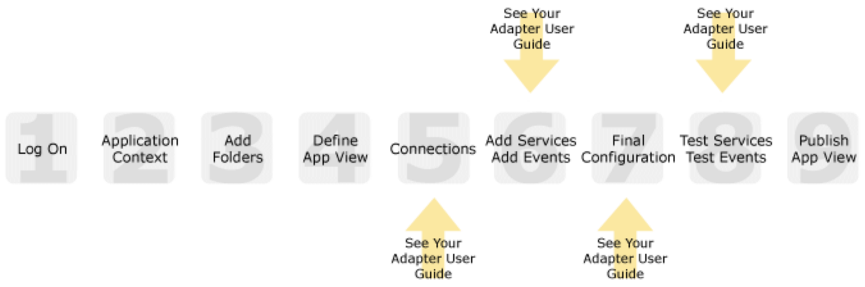
This document is designed to be used in conjunction with *Using the Application Integration Design Console*, available at the following URL:

http://edocs.bea.com/wli/docs81/aiuser/index.html

*Using the Application Integration Design Console* descibes, in detail, the process of defining an application view, which is a key part of making an adapter available to process designers and other users. What *Using the Application Integration Design Console* does *not* cover is the specific information about Adapter for Oracle Applications that you need to supply to complete the application view definition. You will find that information in this document.

At each point in *Using the Application Integration Design Console* where you need to refer to this document, you will see a note that directs you to a section in your adapter user guide, with a link to the edocs page for adapters. The following roadmap illustration shows where you need to refer from *Using the Application Integration Design Console* to this document.

**Figure 1 Information Interlock with** *Using the Application Integration Design Console*


See Your Adapter User Guide
See Your Adapter User Guide
1 Log On
2 Application Context
3 Add Folders
4 Define App View
5 Connections
6 Add Services Add Events
7 Final Configuration
8 Test Services Test Events
9 Publish App View
See Your Adapter User Guide
See Your Adapter User Guide

## Contact Us!

Your feedback on the BEA WebLogic Adapter for Oracle Applications documentation is important to us. Send us e-mail at **docsupport@bea.com** if you have questions or comments.

Your comments will be reviewed directly by the BEA professionals who create and update the BEA WebLogic Adapter for Oracle Applications documentation.

In your e-mail message, please indicate that you are using the documentation for BEA WebLogic Adapter for Oracle Applications and the version of the documentation.

If you have any questions about this version of BEA WebLogic Adapter for Oracle Applications, or if you have problems using the BEA WebLogic Adapter for Oracle Applications, contact BEA Customer Support through BEA WebSUPPORT at **www.bea.com**. You can also contact Customer Support by using the contact information provided on the Customer Support Card which is included in the product package.

When contacting Customer Support, be prepared to provide the following information:

- Your name, e-mail address, phone number, and fax number
- Your company name and company address
- Your machine type and authorization codes
- The name and version of the product you are using
- A description of the problem and the content of pertinent error messages

# Documentation Conventions

The following documentation conventions are used throughout this document.

| Convention | Item |
|---|---|
| **boldface text** | Indicates terms defined in the glossary. |
| Ctrl+Tab | Indicates that you must press two or more keys simultaneously. |
| *italics* | Indicates emphasis or book titles. |
| `monospace text` | Indicates code samples, commands and their options, data structures and their members, data types, directories, and file names and their extensions. Monospace text also indicates text that you must enter from the keyboard.<br>*Examples*:<br>`#include <iostream.h> void main ( ) the pointer psz`<br>`chmod u+w *`<br>`\tux\data\ap`<br>`.doc`<br>`tux.doc`<br>`BITMAP`<br>`float` |
| **`monospace boldface text`** | Identifies significant words in code.<br>*Example*:<br>`void` **`commit`** `( )` |
| *`monospace italic text`* | Identifies variables in code.<br>*Example*:<br>`String` *`expr`* |
| UPPERCASE TEXT | Indicates device names, environment variables, and logical operators.<br>*Examples*:<br>LPT1<br>SIGNON<br>OR |
| `{ }` | Indicates a set of choices in a syntax line. The braces themselves should never be typed. |

| Convention | Item |
|---|---|
| `[ ]` | Indicates optional items in a syntax line. The brackets themselves should never be typed. *Example*: `buildobjclient [-v] [-o name ] [-f file-list]... [-l file-list]...` |
| \| | Separates mutually exclusive choices in a syntax line. The symbol itself should never be typed. |
| `...` | Indicates one of the following in a command line: • That an argument can be repeated several times in a command line • That the statement omits additional optional arguments • That you can enter additional parameters, values, or other information The ellipsis itself should never be typed. *Example*: `buildobjclient [-v] [-o name ] [-f file-list]... [-l file-list]...` |
| . . . | Indicates the omission of items from a code example or from a syntax line. The vertical ellipsis itself should never be typed. |

# Introducing the BEA WebLogic Adapter for Oracle Applications

This section introduces the BEA WebLogic Adapter for Oracle Applications and describes how the adapter enables integration with Oracle Applications and WebLogic Integration.

It includes the following topics:

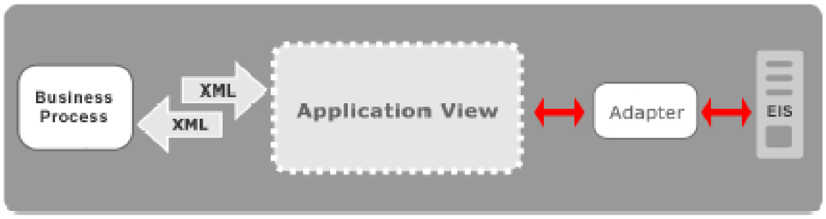
- About Adapters and BEA WebLogic Integration
- Key Components of Integration Solutions
- Getting Started With Adapter for Oracle Applications
- Getting Started With Adapter for Oracle Applications

## About Adapters and BEA WebLogic Integration

The BEA application integration solution uses adapters and application views to help you integrate applications in your enterprise. At its most fundamental level, an *adapter* is software that connects an enterprise information system (EIS) and an integration server. This bi-directional connection consists of *services*—interactions that originate in the adapter (and may require a response from the EIS)—and *events*, interactions that originate in the EIS.

Most EIS systems make selected information and functions available to other applications by way of specialized integration APIs. An adapter connects to the EIS through its integration API, or through database or system calls, and exposes the available functions from the EIS. However, rather than exposing the intricacies of APIs to users, WebLogic Integration incorporates *application views*—business-oriented interfaces that provide a layer of abstraction between an adapter and the EIS capabilities the adapter exposes.

**Figure 1-1 Application View in an Integration Solution**



Application views contain definitions for the services and events used by business processes to communicate with an EIS. They also contain connection information and XML schema that define inputs and outputs for services and events. After an adapter is deployed, you can use its Web-based interface to define as many applications views as you need, and other WebLogic Integration components and applications can use that adapter to access data on the EIS.

To learn more about the role of adapters in application integration architecture, see "Key Components of Integration Solutions" on page 1-3.

To learn more about adapters in general, see the *Introduction to the BEA WebLogic Adapters* at the following URL:

`http://edocs.bea.com/wladapters/docs81/index.html`

## Key Components of Integration Solutions

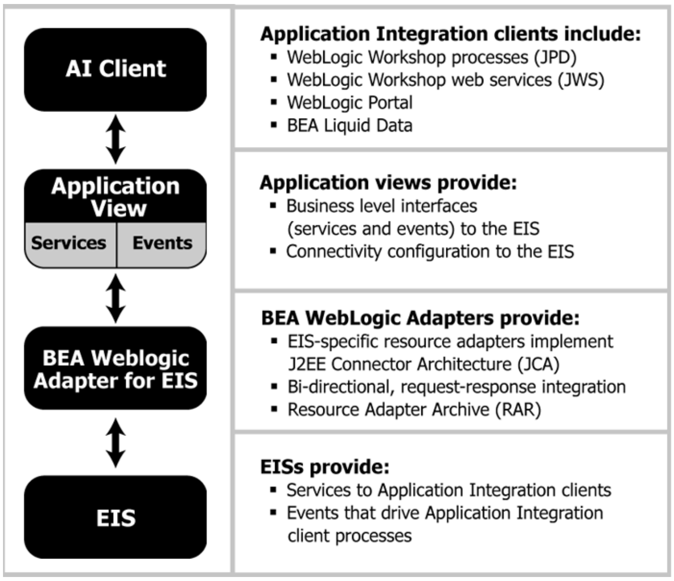
This section describes some of the key concepts you need to be familiar with before you work with an adapter.

- Basic WebLogic Integration Architecture
- Enterprise Information Systems
- Resource Adapters
- Application Views
- Service Clients and Event Consumers
- EIS Metadata, Schemas, and Repositories

### Basic WebLogic Integration Architecture

Adapters are used in conjunction with the Application Integration component of BEA WebLogic Integration. This component provides a systematic, standards-based architecture for hosting business-oriented interfaces to enterprise applications.

**Figure 1-2 Adapters in the Application Integration Architecture**


AI Client
Application View
Services
Events
BEA Weblogic Adapter for EIS
EIS
Application Integration clients include:
WebLogic Workshop processes (JPD)
WebLogic Workshop web services (JWS)
WebLogic Portal
BEA Liquid Data
Application views provide:
Business level interfaces (services and events) to the EIS
Connectivity configuration to the EIS
BEA WebLogic Adapters provide:
EIS-specific resource adapters implement J2EE Connector Architecture (JCA)
Bi-directional, request-response integration
Resource Adapter Archive (RAR)
EISs provide:
Services to Application Integration clients
Events that drive Application Integration client processes

For general information about Application Integration, see the following documents:

- *Introducing Application Integration* at the following URL:

  http://edocs.bea.com/wli/docs81/aiover/index.html

- *Using the Application Integration Design Console* at the following URL:

  http://edocs.bea.com/wli/docs81/aiuser/index.html

## Enterprise Information Systems

An *enterprise information system* (EIS) is software that provides the information infrastructure for an enterprise. An EIS offers a set of services to its clients, which are made available to clients via local and/or remote interfaces. An integration solution involves integration with one or more EISs.

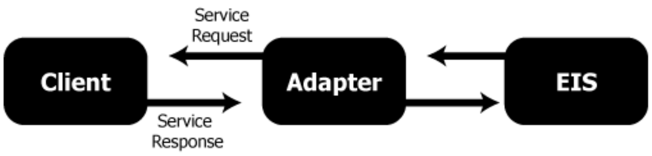
## Resource Adapters

A *resource adapter* (or simply *adapter*) is a BEA software component that acts as a connector between an EIS and a J2EE application server (such as BEA WebLogic Server). Each adapter provides bi-directional, request-response integration with a specific application or technology.

Adapters handle two general types of operations:

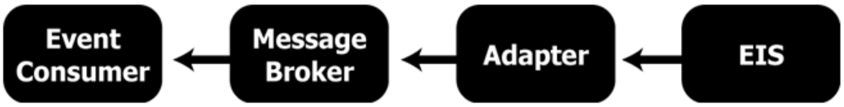
- *Services* are request / response communications with the EIS. Client applications submit service requests to the EIS via the adapter, and the adapter returns the response. For example, a business process might invoke a SAP BAPI or execute a SELECT statement on a database. EIS response back to the client. Responses are either synchronous or asynchronous.

**Figure 1-3 Service Invocations**



- *Events* are asynchronous, one-way messages received from an EIS. For example, the adapter can receive an IDOC from a SAP system or a message from an MQSeries system. The adapter routes the EIS message to the appropriate software component via the WebLogic Integration Message Broker and the Application Integration JMS infrastructure.

**Figure 1-4 Event Notifications**



In effect, a service is a *request for some work to be done* and an event is a *notification that some work has been done*.

For more information about the specific services and events supported by Adapter for Oracle Applications, see "About the BEA WebLogic Adapter for Oracle Applications" on page 1-9.

To learn more about the WebLogic Integration Message Broker and the Application Integration JMS infrastructure, see *Managing WebLogic Integration Solutions* at the following URL: http://e-docs.bea.com/wli/docs81/manage/msgbroker.html.

## Application Views

An *application view* is a business oriented interface to objects and operations within an EIS. Application views include the information needed to communicate with the EIS as well as configurations for services and events.

To learn more about using application view controls in business processes, see Using Integration Controls in the the WebLogic Workshop online help documentation at the following URL:

`http://edocs.bea.com/workshop/docs81/doc/en/integration/controls/controlsIntro.html`

You typically define an application view for a specific business process. Therefore, you can have multiple application views defined for a single adapter, each designed to meet a specific requirement.

An application view defines:

- **Communication with the EIS**, including connection settings, login credentials, and so on.
- **Service invocations**, including the information that the EIS requires for the request, as well as any service request and response schemas associated with the service.
- **Event notifications**, including the information that the EIS publishes and the event schemas for inbound messages.

You create application views in either of two ways:

- For detailed information about application views, see "Understanding Application Views" in Understanding Application Integration in *Introducing Application Integration* at the following URL:

  `http://edocs.bea.com/wli/docs81/aiover/2intfra.html`

- Writing custom code. For more information, see “Using Application Views by Writing Custom Code” in *Using the Application Integration Design Console* at the following URL:

  `http://edocs.bea.com/wli/docs81/aiuser/4usrcust.html`

An application view for Adapter for Oracle Applications provides these features:

- Standards-based data representation. All events, requests, and responses are represented as standards-based XML.
- Abstraction from the details of the EIS. Application views offer a level of abstraction from the details of the underlying EIS, freeing the developers to concentrate on the business processes and data and not on the configuration and details of that system.

To learn more about application views, see Chapter 3, “Defining Application Views for Oracle Applications.”

## Service Clients and Event Consumers

In an integration solution, there are clients that invoke services and consumers for event notifications.

### Service Clients

A variety of clients can invoke services on an EIS via an application view. They include BEA WebLogic Workshop business processes, web services, and portals; queries and BEA Liquid Data; and custom Java applications.

For more information, see the following topics in the BEA WebLogic Workshop Help System:

- “Building Integration Applications”
- “Building Web Services”
- “Building Portal Applications”

at the following URL:

`http://edocs.bea.com/workshop/docs81/doc/en/core/index.html`

In addition, see “Using Applications With Business Processes” in *Using the Application Integration Design Console* at the following URL:

`http://edocs.bea.com/wli/docs81/aiuser/3usruse.html`

### Event Consumers

Adapters deliver events using the WebLogic Integration Message Broker, which provides business processes with a channels-based publish and subscribe communication mechanism. Consumers can include BEA WebLogic Workshop business processes, web services, and portals, as well as custom Java applications.

For more information, see “Message Broker Subscription Control” in Message Broker Controls in Using Integration Controls at the following URL:

`http://edocs.bea.com/workshop/docs81/doc/en/integration/navIntegration.html`

In addition, see “Receiving Events” in “Using Application Views With Business Processes” in *Using the Application Integration Design Console* at the following URL:

`http://edocs.bea.com/wli/docs81/aiuser/3usruse.html`

## EIS Metadata, Schemas, and Repositories

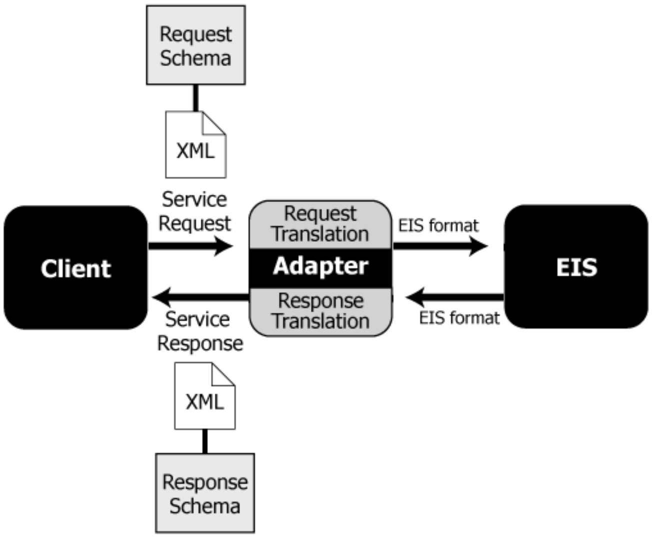
Each EIS uses its own interface to handle service requests and event notifications. Some interfaces are API-based, while others use database or system calls. For example, SAP provides a BAPI interface that defines the parameters and syntax for BAPI requests and responses. For each EIS, the EIS interface defines the *metadata* that applications can use to integrate with the EIS. The EIS publishes data and expects requests in the format dictated by its interface rules and metadata.

### Schema

At run time, the EIS and the adapter exchange service requests, service responses, and events via XML documents. The adapter handles the data translation between XML documents and the EIS format, using *schemas* that map the data between XML and the EIS format: Depending on the adapter, you either create the schema using the BEA Application Explorer, or have it automatically generated by the adapter when you add services and events to an application view.

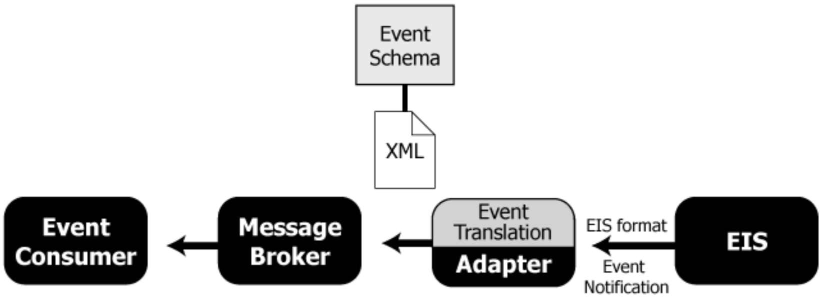
- For service requests, the request arrives at the adapter in the form of an XML document. The adapter uses the *request schema* associated with the service to translate the request to the format that the EIS expects. Similarly, when the adapter receives the response back from the EIS, it uses the *response schema* associated with the service to translate the response to an XML document that the requesting application handles.

**Figure 1-5 Adapter Translation for Service Invocations**



- For event notifications, the message arrives at the adapter in the format that the EIS uses to publish the event. The adapter uses the *event schema* associated with the event to translate the response to an XML document that the subscribed application handles.

**Figure 1-6 Adapter Translation for Event Notifications**



To learn more about schemas, see Chapter 2, "Generating Schemas for Oracle Applications Integration."

### Repositories

Once you have created the necessary schemas, you save them in a file-based *repository*, along with a manifest file that associates the schemas with events and services. When you configure application views in the Application View Console, you specify the location of the repository so that the application view can find the schemas as needed. For more information, see Chapter 3, "Defining Application Views for Oracle Applications."

## About the BEA WebLogic Adapter for Oracle Applications

The BEA WebLogic Adapter for Oracle Applications connects to your Oracle Applications system so that you can easily use your Oracle Applications data and functions within your business processes. The adapter provides scalable, reliable, and secure access to your Oracle Applications system.

This section includes the following topics:

- Oracle Applications Enterprise Application Integration Architecture
- Supported Oracle Applications Operations for Application Integration
- Benefits

### Oracle Applications Enterprise Application Integration Architecture

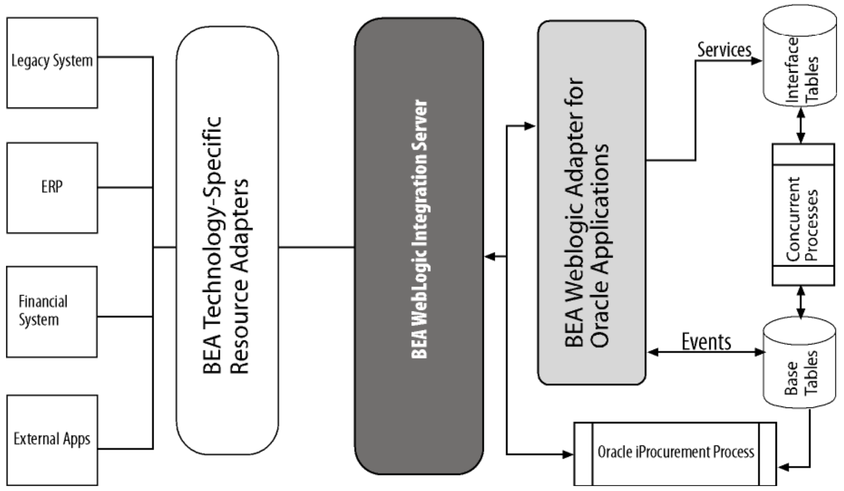
The adapter enables integration with Oracle Applications using the Oracle Applications Enterprise Application Integration (EAI) framework and the Business Integration Manager facility. The adapter uses several methods to access Oracle Applications data and functionality. Choose the methods appropriate to your particular needs.

The adapter supports these methods for integration with Oracle Applications:

- Oracle Advanced Queuing
- Oracle Applications Concurrent Processes
- Oracle Interface Tables
- Base Table access

For a list of the open Oracle Applications interface tables, see Appendix A, "Oracle Interface Tables."

Figure 1-7 Oracle Applications Integration Architecture


Legacy System
ERP
Financial System
External Apps
BEA Technology-Specific Resource Adapters
BEA WebLogic Integration Server
BEA Weblogic Adapter for Oracle Applications
Services
Interface Tables
Concurrent Processes
Events
Base Tables
Oracle iProcurement Process

## Supported Oracle Applications Operations for Application Integration

The Adapter for Oracle Applications supports synchronous and asynchronous, bi-directional message interactions for Oracle Applications.

It provides integration with the following Oracle Applications operations:

- Access to Oracle Applications tables using XML to handle both services and events
- Oracle concurrent programming. The adapter supports access to Oracle interface tables and Advanced Queuing (AQ) queues by both services and events.
- Event reception using Open Applications Group Specification (OAGIS)-formatted XML, commonly known as OAG BODs. The iiProcurement Connector component of Oracle Applications generates these documents.
- Event generation using OAG BODs. These documents are then processed by the iProcurement Connector for input into Oracle Applications.
- Event generation using the Oracle JDBC driver to update an interface table.

### Benefits

The combination of the adapter and WebLogic Integration supplies everything you need to integrate your business processes and enterprise applications with your Oracle Applications system. The Adapter for Oracle Applications provides these benefits:

- Integration can be achieved without custom coding.
- Business processes can be started by events generated by Oracle Applications.
- Business processes can request and receive data from your Oracle Applications system using services.
- Adapter events and services are standards-based. The adapter services and events provide extensions to the *J2EE Connector Architecture* (JCA) version 1.0 from Sun Microsystems, Inc. For more information, see the Sun JCA page at the following URL:

  `http://java.sun.com/j2ee/connector/`
- Improved performance with trigger-based events. Trigger-based events improve performance over other event types when tables are large. Events that use triggers can also provide more granular information than other event types.
- The adapter and WebLogic Integration solution is scalable. The BEA WebLogic Platform provides clustering, load balancing, and resource pooling for a scalable solution. For more information about scalability, see the following URL:

  `http://e-docs.bea.com/wls/docs81/cluster/index.html`
- The adapter and WebLogic Integration solution benefits from the fault-tolerant features of the BEA WebLogic Platform. For more information about high availability, see the following URL:

  `http://edocs.bea.com/wli/docs81/deploy/index.html`
- The adapter and WebLogic Integration solution is secure, using the security features of the BEA WebLogic Platform and the security of your Oracle Applications system. For more information about security, see the following URL:

  `http://edocs.bea.com/wls/docs81/secintro/index.html`

## Getting Started With Adapter for Oracle Applications

This section gives an overview of how to get started using the BEA WebLogic Adapter for Oracle Applications within the context of an application integration solution. Integration with Oracle Applications involves the following tasks:

- Step 1: Design the Application Integration Solution
- Step 2: Determine the Required Oracle Applications Business Processes
- Step 3: Generate Schemas for Oracle Applications Integration Objects
- Step 4: Define Application Views and Configure Services and Events
- Step 5: Integrate with Other BEA Software Components
- Step 6: Deploy the Solution to the Production Environment

## Step 1: Design the Application Integration Solution

The first step is to design an application integration solution, which includes (but is not limited to) such tasks as:

- Defining the overall scope of application integration.
- Determining the business process(es) to integrate.
- Determining which WebLogic Platform components will be involved in the integration, such as web services or business processes designed in WebLogic Workshop, portals created in WebLogic Portal, and so on.
- Determining which external systems and technologies will be involved in the integration, such as Oracle Applications systems and other EISs.
- Determining which BEA WebLogic Adapters for WebLogic Integration will be required, such as the BEA WebLogic Adapter for Oracle Applications. An application integration solution can involve multiple adapters.

This step involves the expertise of business analysts, system integrators, and EIS specialists (including Oracle Applications specialists). Note that an application integration solution can be part of a larger integration solution.

## Step 2: Determine the Required Oracle Applications Business Processes

Within the larger context of an application integration project, you must determine which specific Oracle Applications business objects, tables, views, and processes are required to handle services and events to support the business processes in the application integration solution.

Factors to consider include (but are not limited to):

- Type of Oracle Applications interface tables, views, and processes used in your Oracle Applications system
- Oracle Applications transactions involved in business processes
- Logins required to access Oracle Applications and perform the required operations
- Whether operations are, from the adapter point of view:
  - services, which notify the Oracle Applications system with a request for action, and, in addition, whether such services should be processed synchronously or asynchronously
  - events, which are notifications from the Oracle Applications system that trigger business processes

This step involves the expertise of Oracle Applications specialists, including analysts and administrators.

## Step 3: Generate Schemas for Oracle Applications Integration Objects

After identifying the Oracle Applications integration objects and business processes required for the application integration solution, you must generate the XML schemas that will be used to exchange data with one or more Oracle Applications systems:

- Services require two XML schemas: one for the Oracle Applications request and another for the Oracle Applications response.
- Events require a single XML schema to handle the data sent by the Oracle Applications system.

You use the BEA Application Explorer tool to generate schemas for Oracle Applications operations. To learn more about schemas, see Chapter 2, "Generating Schemas for Oracle Applications Integration."

## Step 4: Define Application Views and Configure Services and Events

After you create the schemas for your Oracle Applications services or events, you create an application view that provides an XML-based interface between WebLogic Server and a particular Oracle Applications system within your enterprise. If you are accessing multiple

Oracle Applications systems, you define a separate application view for each Oracle Applications system you want to access. To provide different levels of security access (such as "guest" and "administrator"), define a separate application view for each security level.

Once you define an application view, you can configure events and services in that application view that employ the XML schemas that you created in "Step 3: Generate Schemas for Oracle Applications Integration Objects" on page 1-13. To learn more about generating schemas, see Chapter 2, "Generating Schemas for Oracle Applications Integration."

To learn more about defining application views, see Chapter 3, "Defining Application Views for Oracle Applications" in conjunction with *Using the Application Integration Design Console*, at the following URL:

`http://edocs.bea.com/wli/docs81/aiuser/index.html`

## Step 5: Integrate with Other BEA Software Components

Once you have configured and published one or more application views for Oracle Applications integration, you can integrate these application views into other BEA software components, such as business processes or web services created in BEA WebLogic Workshop, or portals built with BEA WebLogic Portal.

For more information, see *Using the Application Integration Design Console*, particularly Chapter 3, "Using Application Views with Business Processes," at the following URL:

`http://edocs.bea.com/wli/docs81/aiuser/3usruse.html`

## Step 6: Deploy the Solution to the Production Environment

After you have designed, built, and tested your application integration solution, you can deploy it into a production environment. The following list describes some of the tasks involved in deploying an application integration:

- Design the deployment.
- Deploy the required components of the BEA WebLogic Platform.
- Install and deploy the BEA WebLogic Adapter for Oracle Applications as described in *BEA WebLogic Adapter for Oracle Applications Installation and Configuration Guide*
- Deploy your application views and schemas for Oracle Applications integration.
- Verify business processes in the production environment.
- Monitor and tune the deployment.

# Generating Schemas for Oracle Applications Integration

The adapter uses XML documents to communicate with your Oracle Applications system's tables for both services and events. The format of these XML documents is determined by schemas you generate using the BEA Application Explorer.

This section explains how to use the BEA Application Explorer to generate schemas. It contains the following topics:

- Before You Begin
- About the BEA Application Explorer
- Starting the BEA Application Explorer
- Setting the Session Path
- Managing Oracle Applications Connections
- Managing Schemas

## Before You Begin

Before you begin to generate schema for the adapter, you must:

- Download and install the BEA Application Explorer software. To learn more, see the *BEA Application Explorer Installation and Configuration Guide* at the following URL:

  `http://edocs.bea.com/wladapters/docs81/index.html`

- Obtain the information necessary to connect to your Oracle Applications system. See Table 2-1 for a description of the parameters the BEA Application Explorer requires to connect to Oracle Applications. Contact your Oracle Applications administrator for this information.

## About the BEA Application Explorer

The BEA Application Explorer allows you to browse your Oracle Applications tables, views, and processes.

The BEA Application Explorer uses intelligence about EISes combined with metadata provided by Oracle Applications to generate the schemas required to build application view services and events.

This section contains the following topics:

- About the Process for Defining Schemas
- Types of Schemas Generated by the BEA Application Explorer

### About the Process for Defining Schemas

The process for defining XML schemas includes the following steps:

1. Starting the BEA Application Explorer.
2. Setting the Session Path.

   The BEA Application Explorer uses this path to create the  directory for the schemas.
3. Creating a New Connection or Using an Existing Connection.
4. Creating Schemas for Services and Creating Schemas for Events.

## Types of Schemas Generated by the BEA Application Explorer

Each service or event the adapter uses must be defined by a schema. The BEA Application Explorer generates XML schemas for:

- Service Requests
- Service Responses
- Events
- Concurrent Programs

### Service Requests

*Service requests* are requests for action that your application makes to your Oracle Applications system. Requests are defined by request schema. As part of the definition, the request schema defines the input parameters required by the Oracle Applications system. The Oracle Applications system responds to the request with a service response.

### Service Responses

*Service responses* are the way the Oracle Applications system responds to a service request. A service response schema defines this service response. Service requests always have corresponding responses.

### Events

*Events* are generated by the Oracle Applications system as a result of activity on that system. You can use these events to trigger an action in your application. For example, the Oracle Applications system may generate an event when customer information is updated. If your application must do something when this happens, your application is a consumer of this event. Events are defined by event schema.

### Concurrent Programs

*Concurrent programs* perform several functions, such as transferring data from an interim table to a production table. The adapter supports this functionality with a concurrent programs agent. The concurrent agent submits the concurrent program to an Oracle concurrent manager, which controls Oracle Applications background processing. If your application uses Oracle concurrent programs, you must gnereate schemas for your concurrent programs agents.

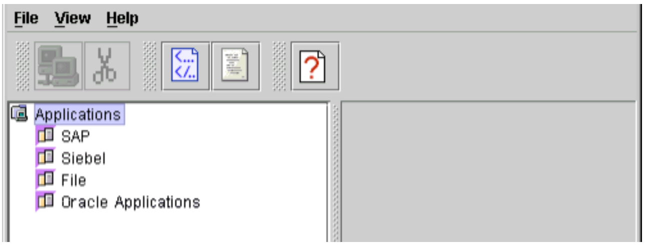
## Starting the BEA Application Explorer

You use the BEA Application Explorer to generate service request schemas, service response schemas, and event schemas. The schemas you create are published in the WebLogic Integration repository.

To start the BEA Application Explorer:

1. Open the BEA Application Explorer.
   - In Windows, choose Windows Start→Programs→BEA Application Explorer.
   - On UNIX, run the startup script `beabse.sh` or the Java command `java com.ibi.common.ui.StartPanel`.

   The BEA Application Explorer window is displayed.



## Setting the Session Path

The session path determines the directory where the BEA Application Explorer places your generated XML schemas and connection information. Your schemas are stored here:

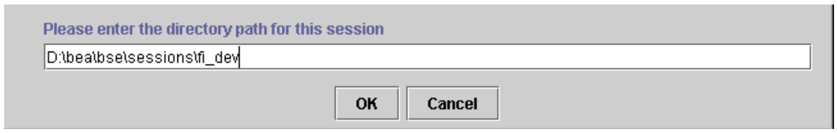
- On Windows: *`session_path`*`\oracleapps\`*`connection_name`*
- On UNIX: *`session_path`*`/oracleapps/`*`connection_name`*

Here, *`connection_name`* is the value you specify when you select a connection. To learn more about selecting a connection, see "Managing Oracle Applications Connections."

To set the session path:

1. From the File menu, choose Session.

   The Enter Session Path window is displayed. It shows a default path.



Please enter the directory path for this session

D:\bea\bse\sessions\fi_dev

2. Do one of the following:
   - To accept the default session path, click OK.
   - To specify a different path, enter the path and click OK.

     Specifying a different path allows you to group your schema according to project, or other logical group.

## Managing Oracle Applications Connections

The BEA Application Explorer must connect to your Oracle Applications system before you can generate schemas. Therefore, you must first define a connection to your Oracle Applications system.

This section includes the following topics:

- About Oracle Applications Logon Parameters
- Creating a New Connection
- Using an Existing Connection
- Disconnecting from Oracle Applications
- Removing Connections

### About Oracle Applications Logon Parameters

In order to create a connection, you must supply valid connection information, including a user name and password for your Oracle Applications system. When creating or editing a connection, the BEA Application Explorer prompts you to provide the Oracle Applications logon parameters. See Table 2-1 for details on these parameters.

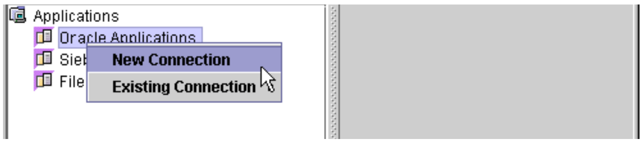
These parameters are used by Oracle Applications client applications to connect to the Oracle Applications system. For more information about these parameters, see your Oracle Applications documentation or consult your Oracle Applications system administrator.

## Creating a New Connection

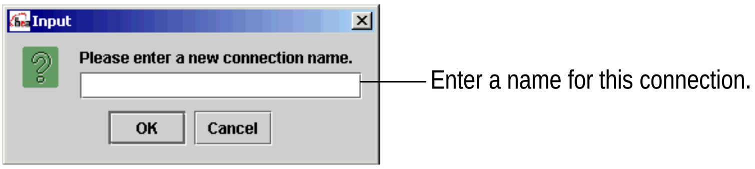
If you are creating a new connection, be sure to check that you have the correct information for your Oracle Applications version for the logon parameters in Table 2-1.

To create a new connection:

1. In the left pane of the BEA Application Explorer window, under Applications right-click Oracle Applications→New Connection.

   The BEA Application Explorer prompts you for a connection name.

   
   Input
   Please enter a new connection name.
   OK
   Cancel
   Enter a name for this connection.

2. Enter a name for this connection and click OK.

   The Oracle Applications Logon dialog box is displayed.

3. Enter the parameters for your system.

   Enter these values:

**Table 2-1 Oracle Applications Logon Parameters**

| Parameter | Description |
|---|---|
| Host | Name or IP address of the machine running Oracle Applications |
| Port | Number of the port on which your Oracle Applications system is listening |
| SID | Database system ID for your Oracle Applications instance |
| User | Valid Oracle Applications user name |
| Password | Password for the Oracle Applications user name |

4. Click OK.

   The new connection is displayed under the Oracle Applications node in the BEA Application Explorer window. You can now browse your Oracle Applications, as well as all available tables, views, and processes in your Oracle Applications system.

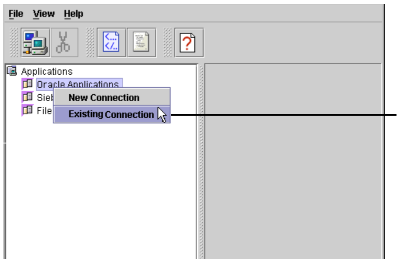
## Using an Existing Connection

You can use an existing connection rather than creating a new one.

To use an existing Oracle Applications connection:

1. In the left pane of the BEA Application Explorer window, under Applications right-click Oracle Applications→Existing Connection→*your connection*.

   The connection is displayed below the Oracle Applications node.



2. If the connection parameters do not correspond to your system, edit them in the Oracle Applications Logon Window. For a description of logon parameters, see "About Oracle Applications Logon Parameters" on page 2-5.

3. Click OK.

## Disconnecting from Oracle Applications

The BEA Application Explorer allows you to disconnect from Oracle Applications.

To disconnect from Oracle Applications:

- In the left pane of the BEA Application Explorer, right-click on the connection. Choose Disconnect.

This disconnects from Oracle Applications, and the connection icon change to indicate that is is not currently connected. To re-establish the connection, right-click on the connection and choose Connect.

### Removing Connections

The BEA Application Explorer allows you to remove connections when you no longer need them. To remove a connection, in the left pane of the BEA Application Explorer, right-click on the connection. Choose Remove.

## Managing Schemas

You need to create a schema for each service and event your application uses. You use the BEA Application Explorer to create these schemas.

This section explains:

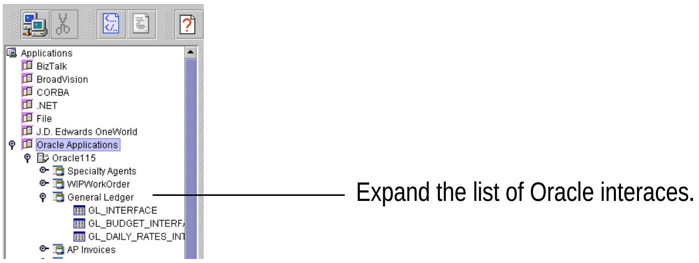
- Creating Schemas for Services
- Creating Schemas for Events
- Creating Schemas for the Concurrent Programs Agent
- Removing Schemas

### Creating Schemas for Services

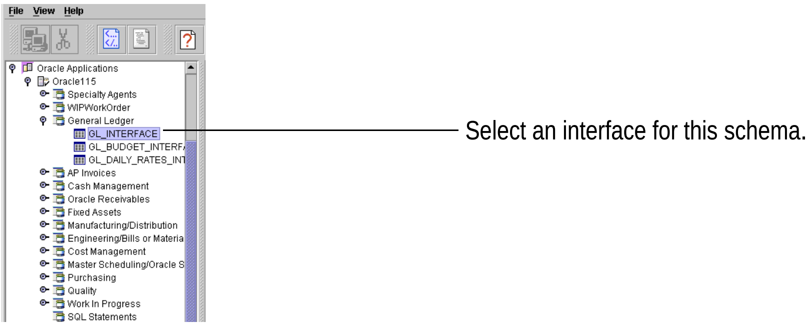
Services require two schemas, one for the request and one for the response. Services always have these two schemas, even if the response is not used by your application.

To create the schemas for a service:

1. Start BEA Application Explorer. To learn more, see "Starting the BEA Application Explorer" on page 2-4.
2. Set the session path. This determines where the BEA Application Explorer places your schemas. To learn more, see "Setting the Session Path" on page 2-4.
3. Select or create a connection to Oracle Applications. To learn more, see "Managing Oracle Applications Connections" on page 2-5.
4. Expand the tree under Applications → Oracle Applications → *connection name* to see the available applications. Select an application. Expand the tree under an application to see the interfaces for which you may create a schema. If you cannot expand the tree beneath Oracle Applications, you have not set a connection for Oracle Applications.

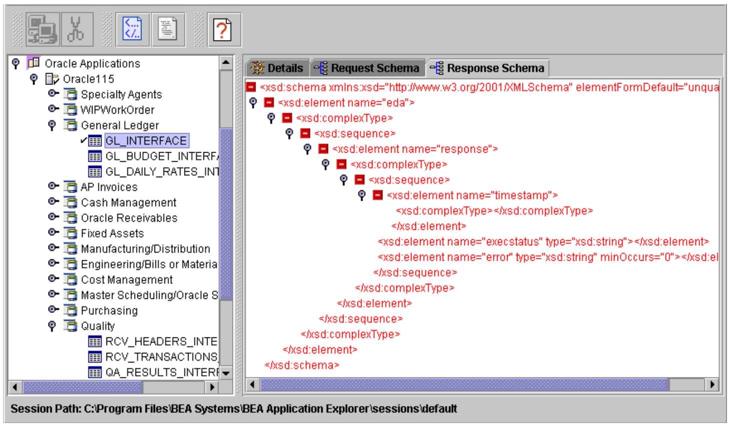


5. Select the interface for this schema. The details of this object appear in the right pane of the BEA Application Explorer window.



6. Right-click the item for which you wish to create the schema and choose Create Service Schemas.

The BEA Application Explorer displays tabs that show the request and response schemas.



```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="unqua
  <xsd:element name="eda">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="response">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="timestamp">
                <xsd:complexType></xsd:complexType>
              </xsd:element>
              <xsd:element name="execstatus" type="xsd:string"></xsd:element>
              <xsd:element name="error" type="xsd:string" minOccurs="0"></xsd:el
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Session Path: C:\Program Files\BEA Systems\BEA Application Explorer\sessions\default

The BEA Application Explorer creates a directory structure within the session path you identified earlier. Note that the session path is indicated in the bottom of the BEA Application Explorer window.

Within this directory, the BEA Application Explorer creates a folder called `oracleapps` as well as subfolders to hold the schemas for each configured Oracle Applications connection. In this example, the schemas have been created in the folder called `Oracle115`, and the BEA Application Explorer adds the following items to the folder:

```
C:\Program Files\BEA Systems\BEA Application Explorer\sessions\default\
oracleapps\Oracle115
```

- `manifest.xml`
- `service_GL_INTERFACE.xsd`
- `service_GL_INTERFACE_response.xsd`

You have successfully created service request and response schemas for this table.

**Note:** The schema names listed above are for the Oracle Financials GL interface. Your schemas will be named for the interfaces you choose.

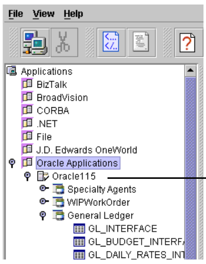
## Creating Schemas for Events

Events only require one schema. There is no response expected when Oracle Applications generate an event.

**Note:** For Oracle Applications, events are supported only for SQL Statements and specialty agents.

To create a schema for an event:

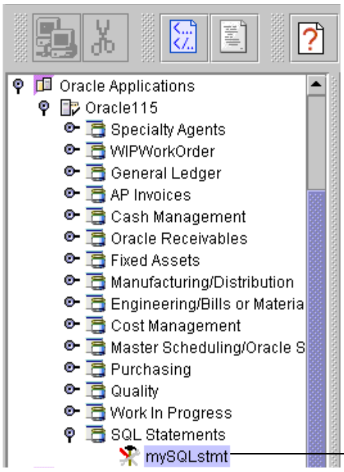
1. Start BEA Application Explorer. To learn more, see "Starting the BEA Application Explorer" on page 2-4.
2. Set the session path. This determines where the BEA Application Explorer places your schemas. To learn more, see "Setting the Session Path" on page 2-4.
3. Select or create a connection to your EIS. To learn more, see "Managing Oracle Applications Connections" on page 2-5.
4. Expand the tree under Applications → Oracle Applications → *connection name* → to see the applications. Select SQL Statements. If you cannot expand the tree beneath Oracle Applications, you have not set a connection for Oracle Applications.



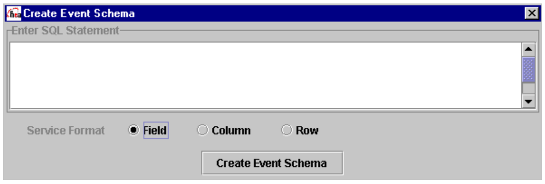
5. Right-click on SQL Statements and choose Add SQL Statement.

6. Enter a name for this SQL statement. Click OK.

   Make a note of this name. You will need it when you create an event in an application view. If you forget the name of the SQL statement you need, you can always start the BEA Application Explorer and see the available SQL statements.



7. Right-click the SQL statement for which you wish to create the schema and choose Create Event Schema. The Enter SQL Statement window is displayed.



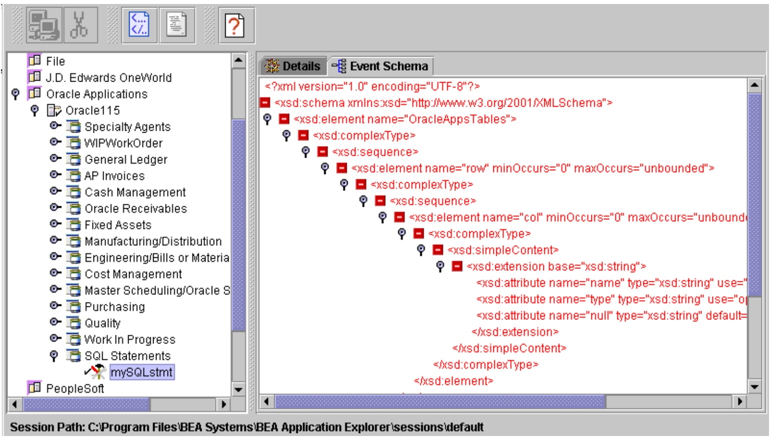
8. Enter in your SQL statement.

The SQL statement you enter here is the input for your event. The resulting data will be sent to your application when the event is triggered.

The adapter supports complex SQL statements, including where clauses and joins.

9. Click Create Event Schema.

   The BEA Application Explorer displays a tab that shows the event schema.



```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="OracleAppsTables">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="row" minOccurs="0" maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="col" minOccurs="0" maxOccurs="unbound
                <xsd:complexType>
                  <xsd:simpleContent>
                    <xsd:extension base="xsd:string">
                      <xsd:attribute name="name" type="xsd:string" use="
                      <xsd:attribute name="type" type="xsd:string" use="o
                      <xsd:attribute name="null" type="xsd:string" default=
                    </xsd:extension>
                  </xsd:simpleContent>
                </xsd:complexType>
              </xsd:element>
```

Session Path: C:\Program Files\BEA Systems\BEA Application Explorer\sessions\default

If it has not already done so, the BEA Application Explorer creates a directory structure within the session path you identified earlier. Note that the session path is indicated in the bottom of the BEA Application Explorer window.

Within this directory, the BEA Application Explorer creates a folder called `oracleapps` as well as subfolders to hold the schemas for each configured Oracle Applications connection. In this example, the schemas have been created in the folder called `Oracle115`, and the BEA Application Explorer adds the following items to the folder:

```
C:\Program Files\BEA Systems\BEA Application Explorer\sessions\default\
oracleapps\Oracle115
```

– `manifest.xml`

– `event_sql0000_mySQLstmt.xsd`

You have successfully created an event schema for this SQL statement.

**Note:** The schema name listed above is for the SQL statement used in this example. Your schemas will be named for the SQL statements or specialty agents you choose.

## Creating Schemas for the Concurrent Programs Agent

For the concurrent programs agent, as for any other service, you must create a request schema and a response schema. For each Oracle Applications connection that you will use to run concurrent programs, you only need to create these schemas *once*. These schemas are sufficient for all the concurrent programs invoked through that connection.

To create the schemas:

1. Start the BEA Application Explorer. To learn more, see "Starting the BEA Application Explorer" on page 2-4.
2. Set the session path. This determines where the BEA Application Explorer places your schemas. To learn more, see "Setting the Session Path" on page 2-4.
3. Select or create a connection to your EIS. To learn more, see "Managing Oracle Applications Connections" on page 2-5.
4. From the left pane of the BEA Application Explorer, select Specialty Agents. This displays the Concurrent Agent node.
5. Right-click Concurrent Agent and select Create Service Schema.

   This creates the service schemas in the session path directory. In the right pane, the Request tab and Response tab appear.

You have finished creating the concurrent program service schema. To learn more about creating schemas, see *Using the Application Integration Design Console* at the following URL:

`http://edocs.bea.com/wli/docs81/aiuser/index.html`

## Removing Schemas

You can remove a schema.

To remove a schema:

1. Right-click on a table for which there is at least one schema.

   If there is an event schema defined for this table, the menu has a Remove Event Schemas option.

   If there are service schemas defined for this table, the menu has a Remove Service Schemas option.
2. Choose the appropriate option.

## Next Steps

After you have defined schemas for your events and services, the next step is to create an application view. An application view makes the services and events available to applications. To learn more about application views, see Defining Application Views for Oracle Applications.

# Defining Application Views for Oracle Applications

An application view is a business-oriented interface to objects and operations within an EIS.

This section presents the following topics:

- How to Use This Document
- Before You Begin
- About Application Views
- About Defining Application Views
- Defining Service Connection Parameters
- Setting Service Properties
- Setting Event Properties
- Defining Event Connection Parameters
- Testing Services
- Testing Events Using a Service
- Testing Events Manually
- Testing the Concurrent Program Agent

## How to Use This Document

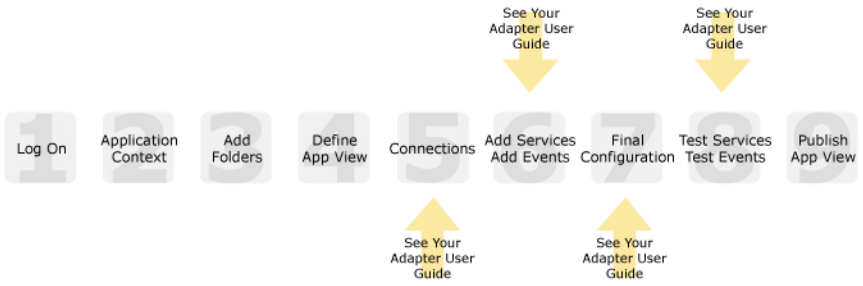
This document is designed to be used in conjunction with *Using the Application Integration Design Console*, available at the following URL:

http://edocs.bea.com/wli/docs81/aiuser/index.html

*Using the Application Integration Design Console* describes, in detail, the process of defining an application view, which is a key part of making an adapter available to process designers and other users. What *Using the Application Integration Design Console* does *not* cover is the specific information—about connections to your Oracle Applications system, as well as supported services and events—that you must supply as part of the application view definition. You will find that information in this section.

At each point in *Using the Application Integration Design Console* where you need to refer to this document, you will see a note that directs you to a section in your adapter user guide, with a link to the edocs page for adapters. The following road map illustration shows where you need to refer from *Using the Application Integration Design Console* to this document.

**Figure 3-1 Information Interlock with** ***Using the Application Integration Design Console***


See Your Adapter User Guide
See Your Adapter User Guide
1 Log On
2 Application Context
3 Add Folders
4 Define App View
5 Connections
6 Add Services Add Events
7 Final Configuration
8 Test Services Test Events
9 Publish App View
See Your Adapter User Guide
See Your Adapter User Guide

## Before You Begin

Before you define an application view, make sure you have:

- Installed and deployed the adapter according to the instructions in *BEA WebLogic Adapter for Oracle Applications Installation and Configuration Guide*.
- Determined which business processes need to be supported by the application view. The required business processes determine the types of services and events you include in your application views. Therefore, you must gather information about the application's business requirements from the business analyst. Once you determine the necessary business

processes, you can define and test the appropriate services and events. For more information, see "Getting Started With Adapter for Oracle Applications" on page 1-11.

- Gathered the connection information for your Oracle Applications system. To learn more about the connection information needed by the BEA Application Explorer for your Oracle Applications system, see "Before You Begin" on page 2-2.

## About Application Views

An application view defines:

- Connection information for the EIS, including login information, connection settings, and so on.
- Service invocations, including the information the EIS requires for this request, as well as the request and response schemas associated with the service.
- Event notifications, including the information the EIS publishes and the event schema for inbound messages.

Typically, an application view is configured for a single business purpose and contains only the services and events required for that purpose. An EIS might have multiple application views, each defined for a different purpose.

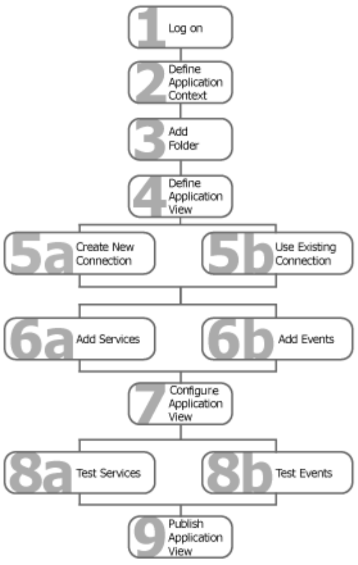
## About Defining Application Views

Defining an application view is a multi-step process described in *Using the Application Integration Design Console*, available at the following URL:

`http://edocs.bea.com/wli/docs81/aiuser/index.html`

The information you enter depends on the requirements of your business process and your EIS system configuration. Figure 3-2 summarizes the procedure for defining and configuring an application view.

**Figure 3-2 Process for Defining and Configuring an Application View**


1 Log on
2 Define Application Context
3 Add Folder
4 Define Application View
5a Create New Connection
5b Use Existing Connection
6a Add Services
6b Add Events
7 Configure Application View
8a Test Services
8b Test Events
9 Publish Application View

To define an application view:

1. Log on to the WebLogic Integration Application View Console.
2. Define the application context by selecting an existing application or specifying a new application name and root directory.

   This application will be using the events and services you define in your application view. The application view works within the context of this application.
3. Add folders as required to help you organize application views.
4. Define a new application view for your adapter.
5. Add a new connection service or select an existing one.

   If you are adding a new connection service, see "Defining Service Connection Parameters" on page 3-5 for details about Oracle Applications requirements.

6. Add the events and services for this application view.

   See the following sections for details about Oracle Applications requirements:

   – "Setting Service Properties" on page 3-7
   – "Setting Event Properties" on page 3-13

7. Perform final configuration tasks.

   If you are adding an event connection, see "Defining Event Connection Parameters" on page 3-19 for details about Oracle Applications requirements.

8. Test all services and events to make sure they can properly interact with the target Oracle Applications system.

   See the following sections for details about Oracle Applications requirements:

   – "Testing Services" on page 3-21
   – "Testing Events Using a Service" on page 3-22
   – "Testing Events Manually" on page 3-23

9. Publish the application view to the target WebLogic Workshop application.

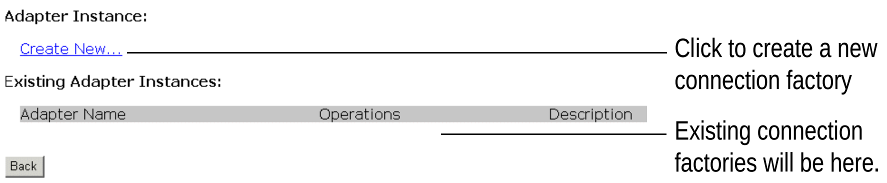
   This is the application you specified in step 2. Business processes in the target application can interact with the newly published application view using the Application View control.

## Defining Service Connection Parameters

This information applies to "Step 5A, Create a New Browsing Connection" in Defining an Application View in *Using the Application Integration Design Console*, at the following URL:

`http://edocs.bea.com/wli/docs81/aiuser/2usrdef.html`

The Select Browsing Connection page allows you to choose the type of connection factory to associate with the application view. You can select a connection factory within an existing instance of the adapter or create a connection factory within a new adapter instance.


Adapter Instance:

Create New...

Existing Adapter Instances:

| Adapter Name | Operations | Description |
|---|---|---|

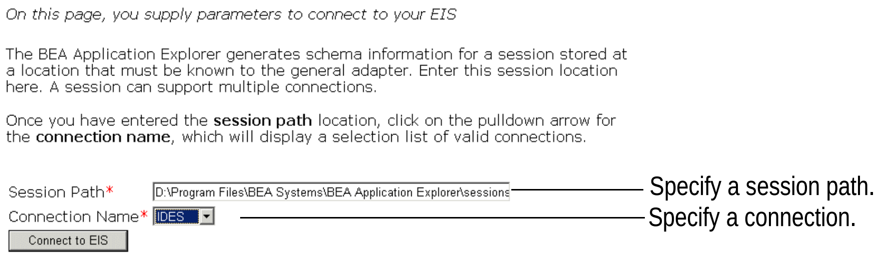
Back
Click to create a new connection factory
Existing connection factories will be here.

After you enter a connection name and description, you use the Configure Connection Parameters page to specify connection parameters for a connection factory.

To create a new browsing connection:

1. In the Create New Browsing Connections page, enter a connection name and description as described in *Using the Application Integration Design Console*.

   The Configure Connection Parameters page is displayed. It allows you to configure the newly created connection factory within the new adapter instance.

   
   *On this page, you supply parameters to connect to your EIS*

   The BEA Application Explorer generates schema information for a session stored at a location that must be known to the general adapter. Enter this session location here. A session can support multiple connections.

   Once you have entered the **session path** location, click on the pulldown arrow for the **connection name**, which will display a selection list of valid connections.

   | | |
   |---|---|
   | Session Path* | D:\Program Files\BEA Systems\BEA Application Explorer\sessions |
   | Connection Name* | IDES |

   **Note:** A red asterisk ( * ) indicates that a field is required.

2. Specify a session path and connection name.

   This information enables the application view to interact with the target Oracle Applications system. You need enter this information only once per application view.

3. Click Connect to EIS.

   You return to the Create New Browsing Connections, where you can specify connection pool parameters and logging levels. For more information, see *Using the Application Integration Design Console* at the following URL:

   `http://edocs.bea.com/wli/docs81/aiuser/index.html`

## Setting Service Properties

This information applies to “Step 6A, Add a Service to an Application View”in Defining an Application View in *Using the Application Integration Design Console*, at the following URL:

`http://edocs.bea.com/wli/docs81/aiuser/2usrdef.html`

Adapter for Oracle Applications uses services to make requests of the Oracle Applications system. A service consists of both a request and a response. BEA WebLogic Adapter for Oracle Applications supports the following services:

- Interface Table
- AQService
- Concurrent Programs Agent

### Interface Table

This information applies to “Step 6A, Add a Service to an Application View” in Defining an Application View in *Using the Application Integration Design Console*, at the following URL:

`http://edocs.bea.com/wli/docs81/aiuser/2usrdef.html`

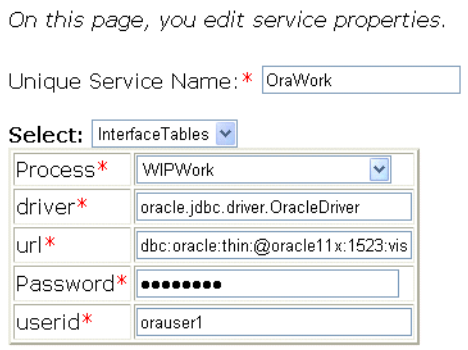
This service uses Oracle interface tables to send data to Oracle Applications.

**Note:** Interface table services require the Oracle JDBC driver. For more information on this driver, see your Oracle DBA or the *BEA WebLogic Adapter for Oracle Applications Installation and Configuration Guide*.

To configure an interface table service:

1. Enter a unique service name that describes the function the service performs.
2. Select InterfaceTables from the Select list.

   The Add Services page displays the fields required for this service type.


*On this page, you edit service properties.*

| | |
|---|---|
| Unique Service Name:* | OraWork |
| Select: | InterfaceTables |
| Process* | WIPWork |
| driver* | oracle.jdbc.driver.OracleDriver |
| url* | dbc:oracle:thin:@oracle11x:1523:vis |
| Password* | ●●●●●●●●● |
| userid* | orauser1 |

**Note:** A red asterisk ( * ) indicates that a field is required.

3. Enter the following information:

**Table 3-1 Interface Table Service Parameters**

| Parameter | Description |
|---|---|
| process | The name of the transaction process. For a list of supported interface tables, see Appendix A, "Oracle Interface Tables." |
| driver | The name of the Oracle JDBC driver. Consult your Oracle Applications DBA for this information. |
| url | The URL to access the Oracle Applications database. Consult your Oracle Applications DBA for this information. |
| password | The password associated with the user ID |
| user | The user ID used to access the Oracle Applications database |

4. See "Common Service and Event Settings" on page 3-12 for information about selecting a schema and configuring logging and tracing.

## AQService

This information applies to "Step 6A, Add a Service to an Application View"in Defining an Application View in *Using the Application Integration Design Console*, at the following URL:

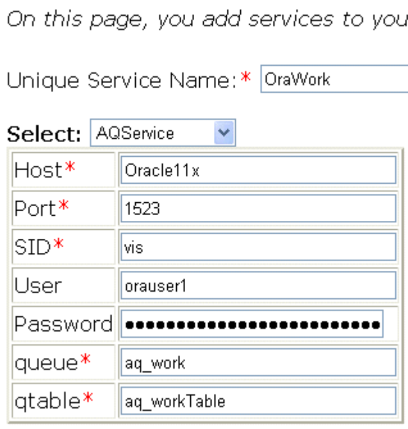
http://edocs.bea.com/wli/docs81/aiuser/2usrdef.html

This service uses Oracle Advanced Queuing (AQ) to send data to Oracle Applications.

To configure an AQService:

1. Enter a unique service name that describes the function the service performs.
2. Select AQService from the Select list.

   The Add Services page displays the fields required for this service type.



*On this page, you add services to your application view.*

| | |
|---|---|
| Unique Service Name:* | OraWork |
| Select: | AQService |
| Host* | Oracle11x |
| Port* | 1523 |
| SID* | vis |
| User | orauser1 |
| Password | •••••••••••••••••••••••••• |
| queue* | aq_work |
| qtable* | aq_workTable |

**Note:** A red asterisk ( * ) indicates that a field is required.

3. Enter the following information.

**Table 3-2 AQService Parameters**

| Parameter | Description |
|---|---|
| host | The machine name or IP address of the machine running Oracle Applications |
| port | The number of the port the Oracle Applications database is listening on |
| SID | The database system ID for the Oracle Applications database. Consult your Oracle Applications DBA for this information. |
| user | The user ID used to access the Oracle Applications database |
| password | The password associated with the user ID |

Table 3-2 AQService Parameters (Continued)

| Parameter | Description |
|---|---|
| queue | The Oracle Advanced Queuing queue name |
| qtable | The table name on which the specified Oracle Advanced Queue resides |

4. See "Common Service and Event Settings" on page 3-12 for information about selecting a schema and configuring logging and tracing.

To test the Concurrent Programs service, see "Testing the Concurrent Program Agent" on page 3-23.

## Concurrent Programs Agent

This information applies to "Step 6A, Add a Service to an Application View" in Defining an Application View in *Using the Application Integration Design Console*, at the following URL:

`http://edocs.bea.com/wli/docs81/aiuser/2usrdef.html`

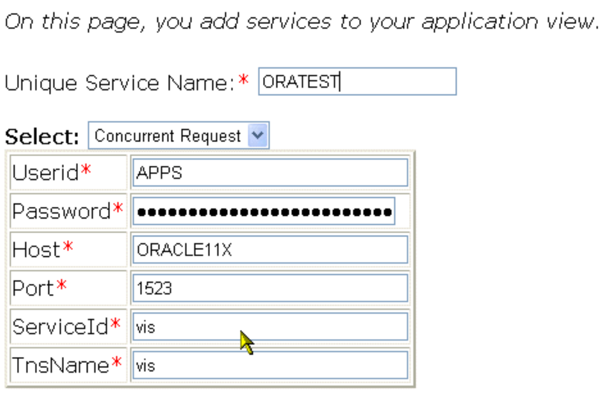
Adding a service for the concurrent programs agent is similar to configuring any other Oracle Applications service.

**Note:** To use a concurrent program service, you must have the concurrent service agent installed. To learn more about installing the concurrent programs agent, see *BEA WebLogic Adapter for Oracle Applications Installation and Configuration Guide*.

To configure a service for a concurrent programs agent:

1. Enter a unique service name that describes the function the service performs.
2. Select ConcurrentRequest from the Select list.

   The Add Services page displays the fields required for this service type.


*On this page, you add services to your application view.*

| | |
|---|---|
| Unique Service Name:* | ORATEST |
| Select: | Concurrent Request |
| Userid* | APPS |
| Password* | ••••••••••••••••••••••••• |
| Host* | ORACLE11X |
| Port* | 1523 |
| ServiceId* | vis |
| TnsName* | vis |

**Note:** A red asterisk ( * ) indicates that a field is required.

3. Enter the following information:

**Table 3. Service Properties for Concurrent Programs Agent**

| Parameter | Definition |
|---|---|
| Userid | The user ID for the Oracle Applications database. This must be `APPS`. |
| Password | The password for the `APPS` Oracle user ID |
| Host | The name of the server running your Oracle Applications instance |
| Port | The number of the port on which the database is listening |
| ServiceId | The unique name that identifies the database. Contact your Oracle Applications DBA for this information. |
| TnsName | The logical name of the Oracle Applications database instance. Contact your Oracle Applications DBA for this information. |

4. See "Common Service and Event Settings" on page 3-12 for information about selecting a schema and configuring logging and tracing.

## Common Service and Event Settings

This information applies to "Step 6A, Add a Service to an Application View" in Defining an Application View in *Using the Application Integration Design Console*, at the following URL:

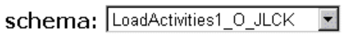
`http://edocs.bea.com/wli/docs81/aiuser/2usrdef.html`

You select a schema and select logging options the same way for all services.

To set common service settings:

1. In the Schema list, select the schema you want to use with this service.

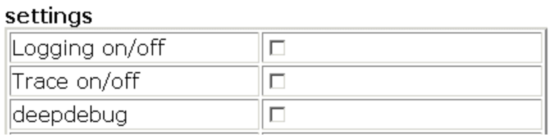
   For more information, see Chapter 2, "Generating Schemas for Oracle Applications Integration."

   schema: LoadActivities1_O_JLCK

2. Configure logging and tracing for this service, as follows:

   Logging captures information from your adapter and writes it in a log file. Tracing displays runtime information in the console. You set the type and amount of information you wish to capture as part of the final configuration tasks. This is described in detail in *Using the Application Integration Design Console*.

   settings

   | | |
   |---|---|
   | Logging on/off | ☐ |
   | Trace on/off | ☐ |
   | deepdebug | ☐ |

   a. Select the Logging on/off check box to enable logging for this service. Logging information is written to a log file (`BEA_ORACLEAPPS_1_0.log`) that appears in the directory from which the application was started.

   b. Select the Trace on/off check box to enable tracing for this service. Trace information appears in the runtime console.

   c. Select the deepdebug check box to enable additional trace information for deeper troubleshooting.

3. Click Add to add the service.

   For more information about the next step, see *Using the Application Integration Design Console* at the following URL:

   `http://edocs.bea.com/wli/docs81/aiuser/index.html`

## Setting Event Properties

This information applies to “Step 6B, Add an Event to an Application View” in Defining an Application View in *Using the Application Integration Design Console*, at the following URL:

`http://edocs.bea.com/wli/docs81/aiuser/2usrdef.html`

An event defines how your application responds to events generated by Oracle Applications. The Adapter for Oracle Applications supports the following events:

- OracleAppsTables
- OracleAQ

### OracleAppsTables

This information applies to “Step 6B, Add an Event to an Application View” in *Using the Application Integration Design Console*, at the following URL:

`http://edocs.bea.com/wli/docs81/aiuser/index.html`

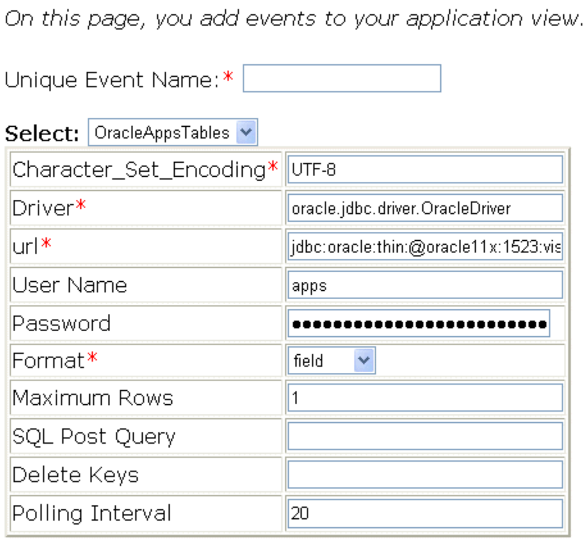
This event uses Oracle interface tables to receive data from Oracle Applications.

**Note:** In order to use an event that uses Oracle Applications interface tables, you must have the Oracle JDBC driver installed. To learn more about installing the Oracle JDBC driver, see your Oracle DBA or the *BEA WebLogic Adapter for Oracle Applications Installation and Configuration Guide*.

To configure an interface table event:

1. Enter a unique event name that describes the function the event performs.
2. Select OracleAppsTables from the Select list.

   The Add Events page displays the fields required for this event type.


*On this page, you add events to your application view.*

| | |
|---|---|
| Unique Event Name:* | |
| Select: | OracleAppsTables |
| Character_Set_Encoding* | UTF-8 |
| Driver* | oracle.jdbc.driver.OracleDriver |
| url* | jdbc:oracle:thin:@oracle11x:1523:vis |
| User Name | apps |
| Password | •••••••••••••••••••••••••• |
| Format* | field |
| Maximum Rows | 1 |
| SQL Post Query | |
| Delete Keys | |
| Polling Interval | 20 |

**Note:** A red asterisk ( * ) indicates that a field is required.

3. Enter the following information:

**Table 3-3 OracleAppsTable Event Parameters**

| Parameter | Description |
|---|---|
| Character Set Encoding | The character set encoding for inbound documents. For example, UTF-8. |
| driver | The name of the Oracle JDBC driver Consult your Oracle Applications DBA for this information. |
| url | The URL to access the Oracle Applications database. Consult your Oracle Applications DBA for this information. |
| user name | The user ID used to access the Oracle Applications database |
| password | The password associated with the user ID |
| format | Either one of the following values: <br>• column <br>• field <br>To learn more about formats, see Using Formats. |

**Table 3-3 OracleAppsTable Event Parameters (Continued)**

| Parameter | Description |
|---|---|
| maximum rows | The maximum number of rows retrieved from the table in a single operation. You should not allow this parameter to exceed the number of parallel threads available for execution. |
| SQL Post Query | One or more SQL statements that are executed after each new record has been read from the table. If you use more then one statement, separate them with a semicolon (;).<br>If you do not specify a SQL statement here, the default statement erases all the table's data after reading that data:<br>`DELETE field1, field2, ... FROM table`<br>To learn more about post queries, see Using SQL Post Queries. |
| delete keys | A comma-separated list of keys used in the default SQL post query statement. The delete statement operates on keys, so indicate the table's key columns.<br>To learn more about post queries, see Using Delete Keys. |
| polling interval | Interval in seconds at which the table is monitored for new rows. The default value is two seconds. |

4. See "Common Service and Event Settings" on page 3-12 for information about selecting a schema and configuring logging and tracing.

## Using Formats

The OracleAppsTable event formats parameter has two possible values:

- column—This value indicates that the data produced by the query is returned field by field. Each of these field is enclosed in <column> tags. The column tag has an attribute whose value is the name of the field; for example,

```
<row>
<column name="ID">1000</column>
<column name="First_Name">Scott</column>
</row>
```

- field—This value indicates that the data produced by the query is returned field by field. Each field is enclosed in a tag that bears the field name; for example,

```
<row>
<ID>1000></column>
<FIRST_NAME>Scott</column>
</row>
```

## Using SQL Post Queries

The OracleAppsTable event parameter SQL Post Query allows you to specify an action to be done after the query completes.

You can choose to retain the table's data once it has been read by specifying a value for this parameter, as shown in the examples that follow.

If the SQL SELECT statement that you specified for this event (when you created the event's schema in the BEA Application Explorer) included a date column or long column, then you must provide a value for either the SQL Post Query or Delete Keys parameter, and the value you provide must not contain a date column or long column.

**Note:** The SQL Post Query and Delete Keys parameters are mutually exclusive. Delete Keys applies to the default DELETE statement, and SQL Post Query overrides the default DELETE statement. Provide a value for one or the other, but not for both.

There are two field operators, `?` and `^`, that you can use in a post-query SQL statement:

- `?`*`field`* is evaluated at run time as *`field = value`*

  The `?` operator is useful in UPDATE statements:

  ```
  UPDATE table WHERE ?field
  ```

  For example, the following statement

  ```
  UPDATE Stock_Prices_Temp WHERE ?RIC
  ```

  might be evaluated at run time as:

  ```
  UPDATE Stock_Prices_Temp WHERE RIC = 'PG'
  ```

- `^`*`fieldname`* is evaluated at run time as *`value`*

  The ^ operator is useful in INSERT statements:

  ```
  INSERT INTO table VALUES (^field1, ^field2, ^field3, ... )
  ```

  For example, the following statement

  ```
  INSERT INTO Stock_Prices_Temp VALUES (^RIC, ^Price, ^Updated)
  ```

  might be evaluated at run time as:

  ```
  INSERT INTO Stock_Prices_Temp VALUES ('PG', 88.62,
  '2003-03-18 16:24:00.0')
  ```

To learn more about using the SQL Post Query parameter, see "Listening for Newly Inserted Rows Using Row Tracking Events."

### Using Delete Keys

The OracleAppsTable event delete keys parameter allows you to specify rows to be deleted from the table after the query completes.

If the SQL SELECT statement that you specified for this event (when you created the event's schema in the BEA Application Explorer) included a date column or long column, then you must provide a value for either the Delete Keys or SQL Post Query parameters, and the value you provide must not contain a date column or long column.

**Note:** The Delete Keys and SQL Post Query parameters are mutually exclusive, as Delete Keys applies to the default DELETE statement, and SQL Post Query overrides the default DELETE statement. Provide a value for one or the other, but not for both.

To learn more about using the delete keys parameter, see "Listening for Newly Inserted Rows Using Row Removal Events," and "Listening for Table Changes Using Trigger-Based Events."

## OracleAQ

This information applies to "Step 6B, Add an Event to an Application View" in Defining an Application View in *Using the Application Integration Design Console*, at the following URL:

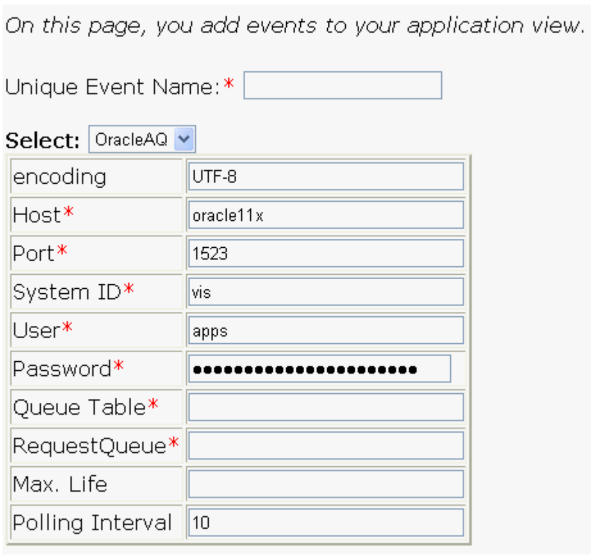
`http://edocs.bea.com/wli/docs81/aiuser/2usrdef.html`

This service uses Oracle Advanced Queuing (AQ) tables to receive data from Oracle Applications.

To configure an OracleAQ event:

1. Enter a unique event name that describes the function the event performs.
2. Select OracleAQ from the Select list.

   The Add Events page displays the fields required for this event type.



*On this page, you add events to your application view.*

|  |  |
|---|---|
| Unique Event Name:* |  |
| Select: | OracleAQ |
| encoding | UTF-8 |
| Host* | oracle11x |
| Port* | 1523 |
| System ID* | vis |
| User* | apps |
| Password* | •••••••••••••••••••••• |
| Queue Table* |  |
| RequestQueue* |  |
| Max. Life |  |
| Polling Interval | 10 |

**Note:** A red asterisk ( * ) indicates that a field is required.

3. Enter the following information:

**Table 3-4 OracleAQ Event Parameters**

| Parameter | Description |
|---|---|
| encoding | The character set encoding for inbound documents, for example, UTF-8 |
| host | The machine name or IP address of the machine running Oracle Applications |
| port | The number of the port the Oracle Applications database is listening on |
| system ID | The database system ID for the Oracle Applications database. Consult your Oracle Applications DBA for this information. |
| user | The user ID used to access the Oracle Applications database |
| password | The password associated with the user ID |
| queue table | The name of the table on which the Oracle Advanced Queuing (AQ) queue resides |
| request queue | The name of the Oracle AQ queue |

**Table 3-4 OracleAQ Event Parameters (Continued)**

| Parameter | Description |
|---|---|
| max life | The maximum time a request may take to complete. This prevents requests that take an inordinately long time. Any request that exceeds this maximum life is terminated. |
| polling interval | Interval in seconds at which the table is monitored for new rows. The default value is two seconds. |

4. See "Common Service and Event Settings" on page 3-12 for information about selecting a schema and configuring logging and tracing.

## Defining Event Connection Parameters

This information applies to "Step 7, Perform Final Configuration Tasks" in Defining an Application View in *Using the Application Integration Design Console*, at the following URL:

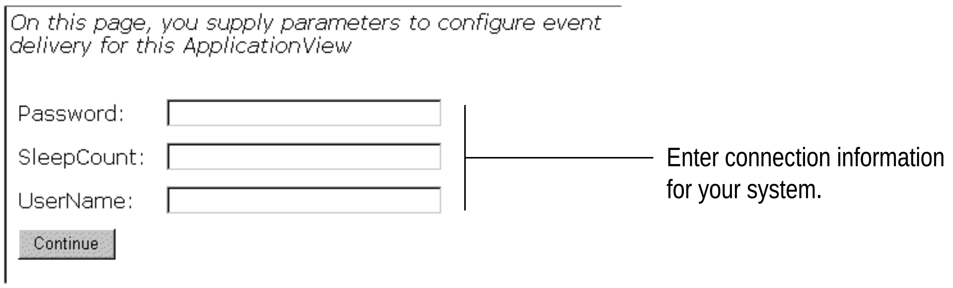
`http://edocs.bea.com/wli/docs81/aiuser/2usrdef.html`

Once you have finished adding services and events and have saved your application view, you must perform some final configuration tasks, including configuring event delivery connections, before testing the services and events. You perform these configuration tasks from the Final Configuration and Testing page.

To define event connection parameters:

1. In Connections area on the Application View Administration page, click Select/Edit.
2. In the Event Connection area, click Event to edit the default event connection.

   The Configure Event Delivery Parameters page is displayed.


On this page, you supply parameters to configure event delivery for this ApplicationView

Password:

SleepCount:

UserName:

**Note:** A red asterisk ( * ) indicates that a field is required.

3. Enter the following information:

**Table 3-5 Event Connection Parameters**

| Parameter | Description |
|---|---|
| username | Your WebLogic Server Administration Console user name, defined in the `startWebLogic` script |
| password | The password for your WebLogic Server Administration Console user name |
| SleepCount | The number of seconds the adapter will wait between polling for events |

The event delivery parameters you enter on this page enable connection to your Oracle Applications system and are used when generating events. The parameters are specific to the associated adapter and are defined in the `wli-ra.xml` file within the base adapter.

4. Click Save to save your event delivery parameter settings. Click Continue to return to the Edit Event Adapter page, and then click Back to return to the Final Configuration and Testing page.

   The Edit Event Adapter page allows you to define event parameters and configure the information that will be logged for the connection factory. Select one of the following settings for the log:

   – Log errors and audit messages

   – Log warnings, errors, and audit messages

   – Log informational, warning, error, and audit messages

   – Log all messages

**Note:** For maximum tracing, select Log all Messages. This is the recommended setting to use when you are collecting debugging information for BEA support.

The table that follows describes the type of information that each logging message contains.

**Table 3-6 Logging message categories**

| This type of message | Contains |
|---|---|
| Audit | Extremely important information related to the business processing performed by an adapter. |
| Error | Information about an error that has occurred in the adapter, which may affect system stability. |
| Warning | Information about a suspicious situation that has occurred. Although this is not an error, it could have an impact on adapter operation. |
| Information | Information about normal adapter operations. |

## Testing Services

This information applies to "Step 8A, Test an Application View's Services" in Defining an Application View in *Using the Application Integration Design Console*, at the following URL:

`http://edocs.bea.com/wli/docs81/aiuser/2usrdef.html`

The purpose of testing an application view service is to evaluate whether that service interacts properly with the target Oracle Applications system. When you test a service, you supply any inputs required to start the service. For the Adapter for Oracle Applications, the input is in the form of a valid XML string that acts as input for the service.

**Note:** You can test an application view only if it is deployed and only if it contains at least one event or service. To learn more about deploying an application view, see *Deploying WebLogic Integration Solutions*.

To test a service:

1. In the Application View Administration page, click the Test link beside the service to be tested.

   The Test Services page is displayed.

2. In the Test Service window, enter the appropriate XML strings.

   See Appendix C, “Samples,” for sample XML code to test a service.

3. Click Test.

   The results appear in the Test Results window.

## Testing Events Using a Service

This information applies to “Step 8B, Test an Application View’s Events” in Defining an Application View in *Using the Application Integration Design Console*, at the following URL:

`http://edocs.bea.com/wli/docs81/aiuser/2usrdef.html`

The purpose of testing an application view event is to make sure that the adapter correctly handles events generated by Oracle Applications. When you test an event, you can trigger the event using a service or manually.

**Note:** You can test an application view only if it is deployed and only if it contains at least one event or service. To learn more about deploying an application view, see *Deploying WebLogic Integration Solutions*.

To test an event:

1. In the Application View Administration page, click the Test link beside the service to be tested.

   The Test Events page is displayed.

2. Click Service and select a service that triggers the event you are testing.

3. In the Time field, enter a reasonable period of time to wait, specified in milliseconds, before the test times out (One second = 1000 milliseconds. One minute = 60,000 milliseconds.).

4. Click Test and enter the XML string needed to trigger the service.

   The service is executed.

   – If the test succeeds, the Test Result page is displayed. It shows the event document, the service input document, and the service output document.

   – If the test fails, the Test Result page displays only a Timed Out message.

## Testing Events Manually

This information applies to “Step 8B, Test an Application View’s Events” in Defining an Application View in *Using the Application Integration Design Console*, at the following URL:

`http://edocs.bea.com/wli/docs81/aiuser/2usrdef.html`

To test an event manually:

1. In the Time field, enter a reasonable period of time to wait, specified in milliseconds, before the test times out (One second = 1000 milliseconds. One minute = 60,000 milliseconds.).
2. Click Test. The test waits for an event to trigger it.
3. In your Oracle Applications system, send XML to the adapter.
4. Using the triggering Oracle Applications application, perform an action that executes the service that, in turn, tests the application view event.
   – If the test succeeds, the Test Result page is displayed. This page displays the event document from the application, the service input document, and the service output document.
   – If the test fails or takes too long, the Test Result page is displayed, showing a Timed Out message.

## Testing the Concurrent Program Agent

Once the concurrent program agent is installed and configured, you can test it to make sure that you have correctly configured the service associated with the agent.

The steps to test the concurrent program agent are:

1. Determine the parameters you need to create the XML request document. To learn more, see *Determining the Request Parameters* on page 3-24.
2. Create the request document. To learn more, see *Creating the Request Document* on page 3-27.
3. Submit the request document. To learn more, see *Submitting the Request and Viewing the Response* on page 3-30.

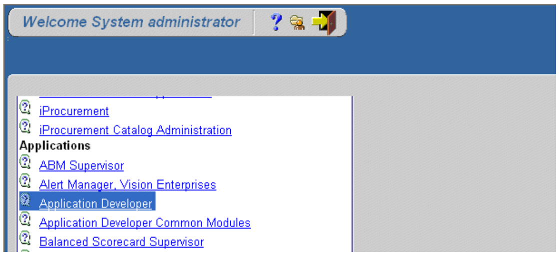
## Determining the Request Parameters

In order to submit a request to a concurrent program, you must include the parameters the program requires. You can use the Oracle Applications Web Client to determine what parameters the event requires.

To determine the parameters:

1. Start the Oracle Applications Web Client. Log on to the Oracle Applications instance on which you wish to run the concurrent program.
2. From the client's main window, select Application Developer.

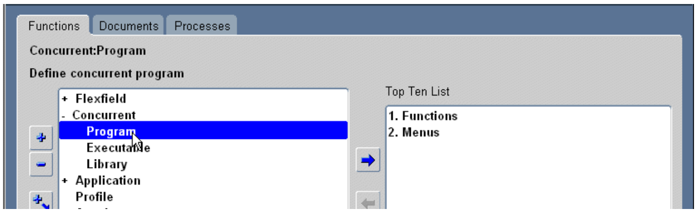
   If you do not see the Application Developer option, contact your system administrator to make sure the account ID you used to log on to the system is enabled.

   Welcome System administrator

   iProcurement
   iProcurement Catalog Administration
   Applications
   ABM Supervisor
   Alert Manager, Vision Enterprises
   Application Developer
   Application Developer Common Modules
   Balanced Scorecard Supervisor

   The Application Developer window is displayed.
3. Expand the Concurrent node and double-click Program.

   Functions | Documents | Processes

   Concurrent:Program

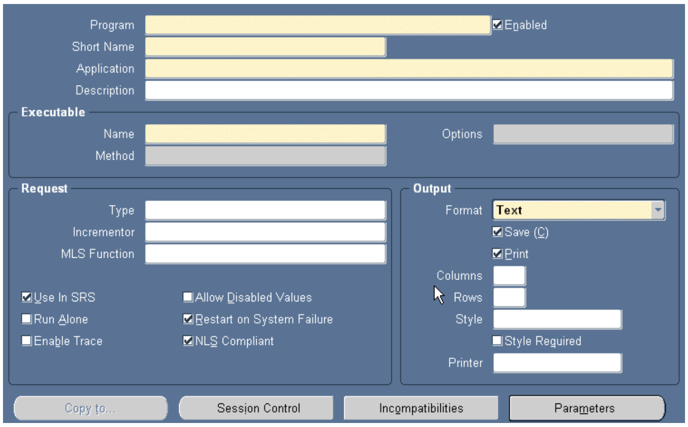
   Define concurrent program

   + Flexfield
   - Concurrent
     Program
     Executable
     Library
   + Application
   Profile

   Top Ten List

   1. Functions
   2. Menus

The Concurrent Programs window is displayed.



4. From the Web Client menu bar, select View→Query By Example→Enter.

   To show that the form is in query mode, the fields in the Web Client's Concurrent Programs window turn blue.

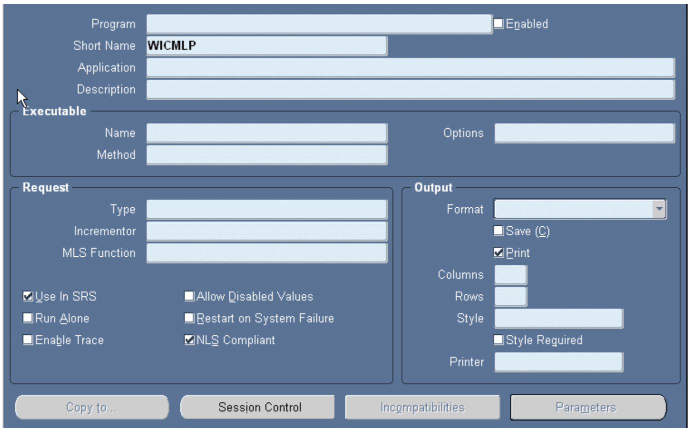
5. Search for the record that contains the name or description of the concurrent program you wish to run.

   – To search on the name, enter the concurrent program's short name in the Short Name field. For example, if your system has a concurrent program named WICMLP, enter that in the Short Name field.

     The names of the concurrent programs available for your use depend on your Oracle Applications installtion.

   – To search on the description, enter the description in the description field.

This example shows a search for the WICMLP for Work In Process (WIP) Mass Load concurrent program.



Program: 
[ ] Enabled
Short Name: WICMLP
Application: 
Description:

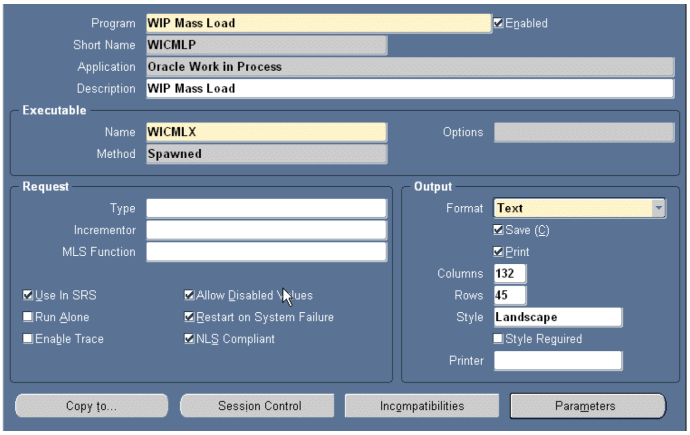
**Executable**
Name: 
Method: 
Options:

**Request**
Type: 
Incrementor: 
MLS Function: 
[✓] Use In SRS
[ ] Run Alone
[ ] Enable Trace
[ ] Allow Disabled Values
[ ] Restart on System Failure
[✓] NLS Compliant

**Output**
Format: 
[ ] Save (C)
[✓] Print
Columns: 
Rows: 
Style: 
[ ] Style Required
Printer:

6. Start the search by selecting View→Query By Example→Run.

   The record is displayed.



Program: WIP Mass Load
[✓] Enabled
Short Name: WICMLP
Application: Oracle Work in Process
Description: WIP Mass Load

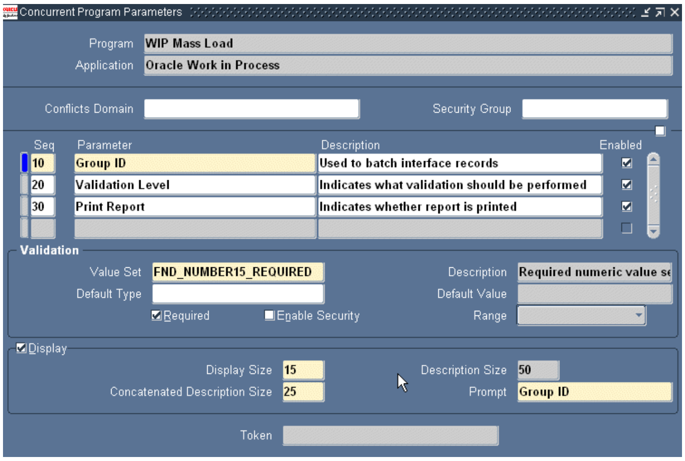
**Executable**
Name: WICMLX
Method: Spawned
Options:

**Request**
Type: 
Incrementor: 
MLS Function: 
[✓] Use In SRS
[ ] Run Alone
[ ] Enable Trace
[✓] Allow Disabled Values
[✓] Restart on System Failure
[✓] NLS Compliant

**Output**
Format: Text
[✓] Save (C)
[✓] Print
Columns: 132
Rows: 45
Style: Landscape
[ ] Style Required
Printer:

7. At the bottom of the Concurrent Programs window, click Parameters .

The window displays the parameters for this concurrent program.



Concurrent Program Parameters

Program: WIP Mass Load
Application: Oracle Work in Process
Conflicts Domain:
Security Group:

| Seq | Parameter | Description | Enabled |
|---|---|---|---|
| 10 | Group ID | Used to batch interface records | ✓ |
| 20 | Validation Level | Indicates what validation should be performed | ✓ |
| 30 | Print Report | Indicates whether report is printed | ✓ |
| | | | ☐ |

Validation
Value Set: FND_NUMBER15_REQUIRED
Description: Required numeric value se
Default Type:
Default Value:
☑ Required ☐ Enable Security
Range:

☑ Display
Display Size: 15
Description Size: 50
Concatenated Description Size: 25
Prompt: Group ID

Token:

8. View a parameter's details by clicking to the left of its sequence number.

9. Make a note of the parameter names and their sequence. For example, in the previous figure, Group ID is first, Validation Level is second, and Print Report is third. You will use this information when you create the request document.

10. Close the Oracle Applications Web Client.

You have finished determining the request document parameters. The next step is to create the request document, as explained in *Creating the Request Document* on page 3-27.

## Creating the Request Document

You must create the XML request document for the concurrent program. The adapter will send this request document to the Oracle concurrent program manager to invoke the concurrent program.

The syntax of the XML request document the concurrent program agent uses to invoke a concurrent program is as follows:

**Listing 7 Concurrent Program Request Document Syntax**

```
<submit_request>
   <resp_appl_shrtnm/>
   <responsibilty/>
   <username/>
   <wait/>
   <prog_appl_shrtnm/>
   <program/>
   <parm/>
   <parm/>
   <parm/>
     .  .  .
   <parm/>
</submit_request>
```

In the preceding listing:

- `resp_appl_shrtnm` is the application short name of your responsibility.

  Valid values for this field depend on your Oracle Applications installation. Generally you can find valid values for this field in the `APPLICATION_SHORT_NAME` column in the `FND_APPLICATION` table.

  The default is `SYSADMIN`.

- `responsibility` is the responsibility of the application user.

  The default is `System Administrator`. Valid values for this field depend on your Oracle Applications installation. Generally you can find valid values for this field in the `RESPONSIBILITY_NAME` column in the `FND_RESPONSIBILITY` and `FND_RESPONSIBILITY_TL` tables.

  **Note:** The `responsibility` and `resp_appl_shrtnm` need not be the same, but they must match. In this example, the `resp_appl_shrtnm` is `SYSADMIN` and the `responsibility` is `System Administrator`.

- `username` is the name of your Application Object Library user.

  This name will be used to update WHO information for data that the concurrent program changes and to create the report output file name for the specific request. This parameter is different from the user ID used to log on to the Oracle database.

  Valid values for this field depend on your Oracle Applications installation. Generally you can find valid values for this field in the `USER_ID` column in the `FND_USER` table.

  The default is `SYSADMIN`.

- `wait` is a boolean indicating whether the request is synchronous or asynchronous.

  This must be Y or N:

  - Y indicates that the request is synchonous and that the agent will check the status of the request will be checked every 60 seconds.
  - N indicates that the request is asynchronous and therefore the status will not be checked. This is the default.

- `prog_appl_shrtnm` is the application short name of your program.

  This value cannot be null.

- `program` is the name of the concurrent program.

  Valid values for this field depend on your Oracle Applications installation. Generally you can find valid values for this field in the `CONCURRENT_PROGRAM_NAME` column in the `FND_CONCURRENT_PROGRAMS` and `FND_CONCURRENT_PROGRAMS_TL` tables.

  This value cannot be null.

  **Note:** The `program` and `prog_appl_shrtnm` need not be the same, but they must match. In this example, the `prog_appl_shrtnm` is `WIP` and the `program` is `WICMLP`.

- `parm` is a concurrent program parameter. There are as many of these as the applicatoin requires. The parameters are distinguished by their place in the sequence.

  To learn more about determining the parameters for your concurrent program and their sequence sequence, see *Determining the Request Parameters* on page 3-24.

The following listing is a request for running the WIP Mass Load example concurrent program.

**Listing 3-1 Request Document for WIP Mass Load Concurrent Program**

```
<submit_request>
   <resp_appl_shrtnm>SYSADMIN</resp_appl_shrtnm>
   <responsibility>System Administrator</responsibility>
   <username>SYSADMIN</username>
   <wait>Y</wait>
   <prog_appl_shrtnm>WIP</prog_appl_shrtnm>
   <program>WICMLP</program>
   <parm>1211</parm>
   <parm>0</parm>
   <parm>1<parm>
</submit_request>
```

In this listing, the parameters are the same as those shown in the Oracle Applications Web Client's Concurrent Program Parameters window in *Determining the Request Parameters* on page 3-24:

- The first paramter is Group ID.
- The second parameter is Validation Level.
- The third parameter is Print Report.

For a complete list of Oracle Concurrent program parameters, see your Oracle documentation.

For a complete list of concurrent program parameters, see your Oracle Applications documentation.

## Submitting the Request and Viewing the Response

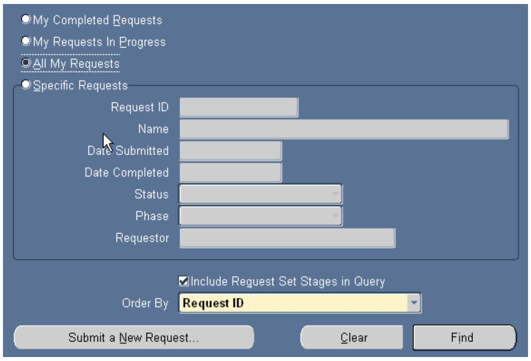
To submit a request document to a concurrent programs agent and view the response:

1. Create the schemas for the concurrent programs agent. To learn more, see *Creating Schemas for the Concurrent Programs Agent* on page 2-13.
2. Add a service for the concurrent programs agent. To learn more, see *Concurrent Programs Agent* on page 3-10.
3. Start the Application View Console, and open the Application View Summary window.
4. In the Services area, locate the service and click Test.

   The Test Service page opens.
5. Enter the XML that invokes the service request. To learn more about constructing the XML request document, see *Determining the Request Parameters* on page 3-24 and *Creating the Request Document* on page 3-27.
6. Click Test to test the service.

   The Results window opens, displaying the request and response documents.
7. Log on to the Oracle Applications instance through the Oracle Applications Web Client.
8. From the main drop-down menu for the desired module, select View→Requests.

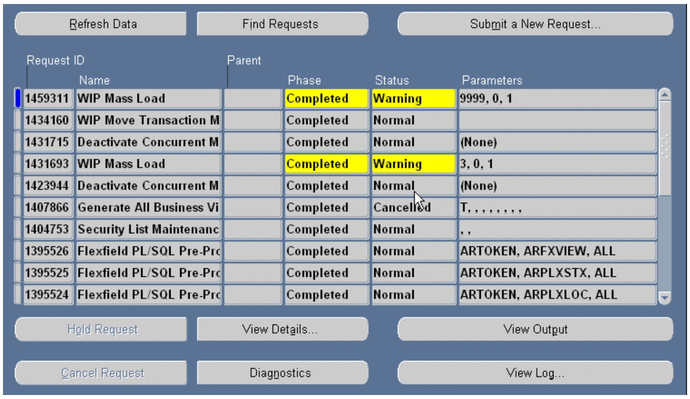
The Find Request window is displayed.



9. If you wish, you can enter selection criteria (for example, Request ID).

10. Click Find.

    The Requests window is displayed.



| Request ID | Name | Parent | Phase | Status | Parameters |
|---|---|---|---|---|---|
| 1459311 | WIP Mass Load | | Completed | Warning | 9999, 0, 1 |
| 1434160 | WIP Move Transaction M | | Completed | Normal | |
| 1431715 | Deactivate Concurrent M | | Completed | Normal | (None) |
| 1431693 | WIP Mass Load | | Completed | Warning | 3, 0, 1 |
| 1423944 | Deactivate Concurrent M | | Completed | Normal | (None) |
| 1407866 | Generate All Business Vi | | Completed | Cancelled | T, , , , , , , , |
| 1404753 | Security List Maintenanc | | Completed | Normal | , , |
| 1395526 | Flexfield PL/SQL Pre-Prc | | Completed | Normal | ARTOKEN, ARFXVIEW, ALL |
| 1395525 | Flexfield PL/SQL Pre-Prc | | Completed | Normal | ARTOKEN, ARPLXSTX, ALL |
| 1395524 | Flexfield PL/SQL Pre-Prc | | Completed | Normal | ARTOKEN, ARPLXLOC, ALL |

The new request is displayed at the top of the list.

Depending on the concurrent program that was executed, one more of the buttons at the bottom of the window may be enabled.  You can click one to view detail information about the concurrent program.

You have finished testing the concurrent program agent.

# Processing Oracle Events

In Chapter 3, "Defining Application Views for Oracle Applications," you learned how to create application view events and set their properties. This section describes different situations in which you listen for events. For each of these situations, this section explains how to use Oracle Applications and the adapter to process these Oracle Applications events.

This section is organized as follows:

- Listening for Newly Inserted Rows Using Row Tracking Events
- Listening for Newly Inserted Rows Using Row Removal Events
- Listening for Table Changes Using Trigger-Based Events

## Listening for Newly Inserted Rows Using Row Tracking Events

Row tracking is the simplest way to listen for events indicating new rows in a table. When you use row tracking, an event listens for newly entered rows in a table. This table is referred to as the *source table*. When there is a new row, the event sends this entire row to a business process. When the event sends a row, it notes this in a control table. The `control table` allows the event to distinguish a new row from one that it has already sent to the business process.

This is the most common way to listen to Oracle Applications events.

### About the Source Table

In order to use row tracing events, the source table must have a column of key values that are unique. These key values must be unique, sortable, and each value must indicate the relative order the row was inserted into the table. This is referred to as an *event key*. A monotonically increasing serial number would be an good event key. A time and datestamp indicating when the row was inserted could be a good event key if the value is granular enough to guarantee uniqueness.

### About the Control Table

You must also create a control table that has one column that is the same type as the event key in the source table. This is where your event writes the event key of the row it has most recently sent to the business process. The event overwrites the single row of this table every time it sends a newly inserted row to your business process. The event determines that a row is newly inserted by comparing the value of its event key against the value of the field in this control table.

### Configuring an Event for Row Tracking

Exactly how you set up your Oracle tables to support row tracking events depends on the configuration of your Oracle Applications system. To learn more about Oracle Applications tables, keys, and data types, see your Oracle documentation.

To configure an event for row tracking:

1. Make sure your table has a column that is suitable as an event key.
2. Create the control table. If necessary, seed it with an appropriate value.
3. In the BEA Application Explorer, create an event schema for a SQL statement. This SQL statement must select from the source table, using the value in the control table to detect newly inserted rows. To learn more about creating an event schema, see Creating Schemas for Events. To learn more about SQL SELECT statements, see your Oracle documentation.
4. In the Application View Console, create an application view that has an event that uses the schema you created for the SQL statement. To learn more about creating an application view for an event, see Setting Event Properties.
5. In the application view, configure the event's SQL Post Query parameter to update the control table with the event key of the newly-sent row.

## Listening for Newly Inserted Rows Using Row Removal Events

Like row tracking events, row removal events listen for new rows inserted into the table. When the event detects a new row, it sends that newly-inserted row to the business process. Unlike row tracking events, row removal events are used in situations when you are not interacting with the table that is the permanent storage for the data. Since the information in the table does not need to be persistent, the event deletes the row as soon as it sends it to the business process. This means that row removal events do not need a control table. If the row is in the table, then by definition it is new.

If your event is listening to a table that is a conduit rather than the permanent storage for your data, then using row removal events is an efficient method to listen for new rows.

### Configuring an Event for Row Removal

To configure an event for row removal:

1. In the BEA Application Explorer, create an event schema for a SQL statement. This SQL statement must select from the source table. To learn more about creating an event schema, see Creating Schemas for Events. To learn more about SQL SELECT statements, see your Oracle documentation.
2. In the Application View Console, create an application view that has an event that uses the schema you created for the SQL statement. To learn more about creating an application view for an event, see Setting Event Properties.
3. In the application view, configure the event's Delete Keys parameter to delete the row from the source table. Specifically, the Delete Keys parameter is a list of keys indicating the rows that the adapter will delete from the source table.

## Listening for Table Changes Using Trigger-Based Events

Trigger-based events are the most complex method of listening for Oracle Applications events. Trigger-based events are appropriate when you need an event to listen to a large joined group of tables. Listening to a large group of joined tables can absorb a lot of computing resources. Using trigger-based events avoids this excessive resource consumption.

An example of a large group of joined tables is a bill of materials. A bill of materials represents a finished item. This finished item is composed of a number of assemblies. These assemblies in turn, are composed of a number of sub-assemblies. From a database perspective, this can become a very complex group of joined tables. Using this example, trigger-based events are appropriate when you have a business process which must perform an action whenever the bill of materials for a given finished product changes.

### About Triggers

Rather than listen to the joined tables, assign triggers to the source tables. Each trigger monitors one source table. Set each trigger to write change information to a control table whenever the source table changes. With this design, your event only has to listen for changes to the control table. You can use a row removal event to send the rows from the control table to your business process.

This technique reduces overhead and increases performance by having the event listen to the small control table, rather than the very large joined table. In addition, the triggers can write information to the control table that describe the change, allowing you to know whether the row was inserted, updated, or deleted.

You can manage your database triggers using a tool such as SQL*Plus. For information on triggers, see your Oracle documentation.

## Sample Trigger

The following is a sample trigger. It fires when any change is made to the WIP_ENTITY_NAME column of the WIP.WIP_ENTITIES table. When this trigger fires, it writes the cahnge information.

**Listing 4-1 Trigger BEA.BEA_PO_CDC_WE_TRG**

```
CREATE OR REPLACE TRIGGER BEA.BEA_PO_CDC_WE_TRG
AFTER INSERT OR DELETE OR UPDATE OF WIP_ENTITY_NAME
ON WIP.WIP_ENTITIES

FOR EACH ROW

BEGIN

IF INSERTING THEN
   INSERT INTO BEA.BEA_PO_CDC
       VALUES (
           :NEW.WIP_ENTITY_ID,
           :NEW.ORGANIZATION_ID,
           'UPDATE');

ELSE
   INSERT INTO BEA.BEA_PO_CDC
       VALUES (
           :OLD.WIP_ENTITY_ID,
           :OLD.ORGANIZATION_ID,
           'UPDATE');
```

```
END IF;

EXCEPTION
   WHEN DUP_VAL_ON_INDEX THEN
        NULL; --- RECORD ALREADY EXISTS
END;
```

## About the Control Table

The design of the control table depends on your Oracle Applications system, and the information you wish to capture with your trigger-based events. At a minimum, your control table has a column for the key of the item being changed. To use the bill of materials example, this would be the key associated with the finished item. You can also store other information in this control table, such as the operation performed on the table (insert, update, or delete), the name of the table that was changed, and even the field that was changed.

If your application requires it, you can have a control table assigned to each source table to store more granular information about the changes.

To learn more about Oracle tables, triggers, and database design, see your Oracle documentation.

## Configuring an Event for Trigger-Based Processing

Exactly how you set up your Oracle tables to support trigger-based events depends on the configuration of your Oracle Applications system. To learn more about Oracle Applications tables and keys, see your Oracle documentation.

To configure an event for trigger-based processing:

1. Create the control table. To learn more about the control table, see About the Control Table.
2. Create triggers and assign them to the source tables. To learn more about the triggers, see About Triggers.
3. In the BEA Application Explorer, create an event schema for a SQL statement. This SQL statement must select from the control table. To learn more about creating an event schema, see Creating Schemas for Events. To learn more about SQL SELECT statements, see your Oracle documentation.
4. In the Application View Console, create an application view that has an event that uses the schema you created for the SQL statement. To learn more about creating an application view for an event, see Setting Event Properties.

5. In the application view, configure the event's Delete Keys parameter to delete the row from the source table. Specifically, the Delete Keys parameter is a list of keys indicating the rows that the adapter will delete from the control table.

# Oracle Interface Tables

This section includes a listing of the open Oracle Applications interface tables and their associated modules.

It includes tables describing the tables for these applications:

- Oracle Financials
- Oracle Manufacturing/Distribution

## Oracle Financials

**Table A-1 Oracle General Ledger**

| Interface Name | Data Flow Direction | Table, View, or Process | Table, View, or Module Name |
|---|---|---|---|
| JournalEntries | Inbound | Table | GL_INTERFACE |
| BudgetAmounts | Inbound | Table | BUDGET_INTERFACE |
| DailyRates | Inbound | Table | GL_DAILY_RATES_INTERFACE |

**Table A-2 Oracle Accounts Payable**

| Interface Name | Data Flow Direction | Table, View, or Process | Table, View, or Module Name |
|---|---|---|---|
| VendorInvoices (from Invoice Gateway, EDI Gateway, other sources | Inbound | Table | AP_INVOICES_INTERFACE<br>AP_INVOICE_LINES_INTERFACE |
| CreditCardTransactions | Inbound | Table | AP_EXPENSE_FEED_LINES |

**Table A-3 Oracle Cash Management**

| Interface Name | Data Flow Direction | Table, View, or Process | Table, View, or Module Name |
|---|---|---|---|
| BankStatements | Inbound | Table | CE_STATEMENT_HEADERS_INT_ALL<br>CE_STATEMENT_LINES_INTERFACE |

**Table A-4 Oracle Receivables**

| Interface Name | Data Flow Direction | Table, View, or Process | Table, View, or Module Name |
|---|---|---|---|
| Customers | Inbound | Table | RA_CUSTOMER_INTERFACE<br>RA_CUSTOMER_PROFILES_INTERFACE<br>RA_CONTACT_PHONES_INTERFACE<br>RA_CUSTOMER_BANKS_INTERFACE |
| CustomerInvoices | Inbound | Table | RA_INTERFACE_LINES<br>RA_INTERFACE_SALESCREDITS<br>RA_INTERFACE_DISTRIBUTIONS |
| ReceiptsFromBanks | Inbound | Table | AR_PAYMENTS_INTERFACE_ALL |

**Table A-5 Oracle Fixed Assets**

| Interface Name | Data Flow Direction | Table, View, or Process | Table, View, or Module Name |
|---|---|---|---|
| CreateAssets (mass additions) | Inbound | Table | FA_MASS_ADDITIONS |

## Oracle Manufacturing/Distribution

**Table A-6 Oracle Inventory**

| Interface Name | Data Flow Direction | Table, View, or Process | Table, View, or Module Name |
|---|---|---|---|
| Transactions | Inbound | Table | MTL_TRANSACTIONS_INTERFACE<br>MTL_SERIAL_NUMBERS_INTERFACE<br>MTL_TRANSACTION_LOTS_INTERFACE |
| DemandInterface | Inbound | Table | MTL_DEMAND_INTERFACE |
| OnHandBalances | Outbound | View | MTL_ITEM_QUANTITIES_VIEW |
| UserDefinedSupply | Inbound | Table | MTL_USER_SUPPLY |
| UserDefinedDemand | Inbound | Table | MTL_USER_DEMAND |
| Replenishment | Inbound | Table | MTL_REPLENISH_HEADERS_INT<br>MTL_REPLENISH_LINES_INT |
| Item | Inbound | Table | MTL_SYSTEM_ITEMS_INTERFACE<br>MTL_ITEMS_REVISIONS_INTERFACE |
| CustomerItem | Inbound | Table | MTL_CI_INTERFACE |
| CustItemCrossReferences | Inbound | Table | MTL_CI_XREFS_INTERFACE |

**Table A-7 Oracle Engineering / Oracle Bills of Material**

| Interface Name | Data Flow Direction | Table, View, or Process | Table, View, or Module Name |
|---|---|---|---|
| MFGCalendar | Outbound | View | BOM_CALENDAR_MONTHS_VIEW |
| BillofMaterial | Inbound | Table | BOM_BILL_OF_MTLS_INTERFACE<br>BOM_INVENTORY_COMPS_INTERFACE<br>BOM_REF_DESGS_INTERFACE<br>BOM_SUB_COMPS_INTERFACE<br>MTL_ITEMS_REVISIONS_INTERFACE |
| Routings | Inbound | Table | BOM_OP_ROUTINGS_INTERFACE<br>BOM_OP_SEQUENCES_INTERFACE<br>BOM_OP_RESOURCES_INTERFACE<br>MTL_RTG_ITEM_REVS_INTERFACE |
| ECO (engineering change orders) | Inbound | Table | ENG_ENG_CHANGES_INTERFACE<br>ENG_ECO_REVISIONS_INTERFACE<br>ENG_REVISED_ITEMS_INTERFACE<br>BOM_INVENTORY_COMPS_INTERFACE<br>BOM_REF_DESGS_INTERFACE<br>BOM_SUB_COMPS_INTERFACE |

**Table A-8 Oracle Cost Management**

| Interface Name | Data Flow Direction | Table, View, or Process | Table, View, or Module Name |
|---|---|---|---|
| ItemCostInquiry | Outbound | View | CST_INQUIRY_TYPES |
| MFGCostReporting | Outbound | View | CST_REPORT_TYPES |
| PeriodicCost (only for the 1st opened period) | Inbound | Table | CST_PC_ITEM_COST_INTERFACE<br>CST_PC_COST_DET_INTERFACE |

**Table A-9 Oracle Master Scheduling/MRP and Oracle Supply Chain Planning**

| Interface Name | Data Flow Direction | Table, View, or Process | Table, View, or Module Name |
|---|---|---|---|
| Forecast | Inbound | Table | MRP_FORECAST_INTERFACE |
| MasterSchedule | Inbound | Table | MRP_SCHEDULE_INTERFACE |
| MasterScheduleRelief | Inbound | Table | MRP_RELIEF_INTERFACE |
| PlannerWorkbench | Outbound | Process | Stored Procedure MRPPL06, or WIP_JOB_SCHEDULE_INTERFACE<br>PO_REQUISITIONS_INTERFACE<br>PO_RESCHEDULE_INTERFACE |
| ProjectedRequirements | Outbound | Table | MRP_RECOMMENDATIONS |
| ProjectedSupply | Outbound | Table | MRP_GROSS_REQUIREMENTS |

**Table A-10 Oracle Master Scheduling/MRP and Oracle Supply Chain Planning**

| Interface Name | Data Flow Direction | Table, View, or Process | Table, View, or Module Name |
|---|---|---|---|
| OrderImport | Inbound | Table | SO_HEADERS_INTERFACE_ALL<br>SO_HEADER_ATTRIBUTES_INTERFACE<br>SO_LINES_INTERFACE_ALL<br>SO_LINE_ATTRIBUTES_INTERFACE<br>SO_LINE_DETAILS_INTERFACE<br>SO_SALES_CREDITS_INTERFACE<br>SO_PRICE_ADJUSTMENTS_INTERFACE |
| DeliveryBasedShip<br>Confirm Open Interface | Inbound | Table | WSH_DELIVERIES_INTERFACE<br>WSH_PACKED_CONTAINER_INTERFACE<br>WSH_PICKING_DETAILS_INTERFACE<br>WSH_FREIGHT_CHARGES_INTERFACE |

Table A-11 Oracle Purchasing

| Interface Name | Data Flow Direction | Table, View, or Process | Table, View, or Module Name |
|---|---|---|---|
| Requisitions | Inbound | Table | PO_REQUISITIONS_INTERFACE<br>PO_REQ_DIST_INTERFACE |
| RequisitionReschedule | Inbound | Table | PO_RESCHEDULE_INTERFACE |
| PurchasingDocuments | Inbound | Table | PO_HEADERS_INTERFACE<br>PO_LINES_INTERFACE |
| Receiving | Inbound | Table | RCV_HEADERS_INTERFACE<br>RCV_TRANSACTIONS_INTERFACE |

Table A-12 Oracle Quality

| Interface Name | Data Flow Direction | Table, View or Process | Table, View, or Module Name |
|---|---|---|---|
| CollectionImport | Inbound | Table | QA_RESULTS_INTERFACE |

Table A-13 Oracle Work in Process

| Interface Name | Data Flow Direction | Table, View, or Module Name |
|---|---|---|
| WIPMoveAdapter. | Inbound | WIP_MOVE_TXN_INTERFACE |
| Resource | Inbound | WIP_COST_TXN_INTERFACE |
| WIPWorkOrderAdapter | Inbound | WIP_JOB_SCHEDULE_INTERFACE |
| Material | Inbound | MTL_TRANSACTIONS_INTERFACE<br>MTL_SERIAL_NUMBERS_INTERFACE<br>MTL_TRANSACTION_LOTS_INTERFACE |

**Table A-13 Oracle Work in Process**

| Interface Name | Data Flow Direction | Table, View, or Module Name |
|---|---|---|
| WIPScheduling | Outbound/ Inbound | Stored Procedures<br>WIP_SCHEDULING_INTERFACE |
| Stored Procedures | Inbound | WIP_SCHEDULING_INTERFACE |

# XML Business Functionality in the BEA WebLogic Adapter for Oracle Applications

This section describes the business functionality of the XML supported by the BEA WebLogic Adapter for Oracle Applications. It includes the following topics:

- BEA WebLogic Adapter for Oracle Applications and Oracle Functionality
- XML Business Functionality Descriptions

## BEA WebLogic Adapter for Oracle Applications and Oracle Functionality

The BEA WebLogic Adapter for Oracle Applications supports Oracle Advanced Queuing for WebLogic Integration services and events. This provides direct bidirectional access to the Oracle Internet Procurement Connector for direct Open Application Group (OAG) formatted XML and also to the Oracle XML Gateway, which supports all DTD-based XML standards. The majority of Oracle prebuilt messages delivered with the Oracle e-Business Suite are pre-mapped using the Open Application Group (OAG) standard. Any Oracle prebuilt message may be re-mapped to any standard of choice using the XML Gateway Message Designer.

The Oracle Internet Procurement (OIP) Connector provides for easy integration between the Oracle Internet Procurement application and external systems. The OIP Connector is based on the open, industry-standard Open Application Group Internet Standard (OAGIS) XML. For more information on the OAGIS XML, please see the Open Application Group Web site at `http://www.openapplications.org`. The standard allows OIP Connector interaction with middleware systems adhering to the OAGIS standard.

Oracle XML Gateway is a set of services that allows for easy integration with the Oracle e-Business Suite to support non-standard XML messaging. The Oracle e-Business Suite utilizes the Oracle Workflow Business Event System to publish and subscribe to application business events to automatically trigger the creation and consumption of XML messages.

Oracle XML Gateway consumes events raised by the Oracle e-Business Suite, and subscribes to inbound events for processing. Oracle XML Gateway uses the message propagation feature of Oracle Advanced Queuing to integrate with the Transport Agent to deliver or receive messages to and from business partners.

## XML Business Functionality Descriptions

The following section describes the business functionality of the XML supported by the BEA WebLogic Adapter for Oracle Applications.

### Post Journal

Journal is an XML business document that transmits data necessary to create a journal entry from any sub ledger business application to a general ledger application. This scenario assumes that the details of the financial transactions are kept in the sub ledgers and that the drill back mechanism from the general ledger component to the sub ledger components is addressed by this XML. Many applications in the enterprise environment create data that causes changes in the account balances of a general ledger application (such as Benefits, Costing, Human Resources and Payroll, Inventory, Manufacturing, Production, and Treasury). By no means is this a complete list of all the activities which may generate a journal entry. Many other tasks that occur within the enterprise applications cause the creation of a General Ledger journal entry. For example, the adjustment of inventory value is a task that occurs within Inventory.

Journal supports the summarization of accounting activities from sub ledgers to the general ledger. This summary is used to express account balances, which are used to report on the financial condition of the enterprise. A Journal XML consists of the Journal Entry Header information which includes the type of journal being entered, a Journal Entry Detail Line (typically, there are at least two occurrences given the common accounting rule of every debit requiring an equally balancing credit), and an account code, which consists of a number of separate elements. Some of these elements can be linked to specified XML fields; for example, the nominal account key (GLNOMACCT) or the cost center (COSTCENTER).

## Confirm

The purpose of the Confirm XML is to add a layer of software application to the software application exception capability to the integration solution. The Confirm XML does this by providing a mechanism for the business software layers to communicate to each other in addition to the confirmation mechanisms that may be provided with any middleware tools involved in the solution.

In instances when the receiving application must communicate its native informational messages to an integrated partner application, it is important to ensure that the applications have a mechanism for speaking their own language to each other.

The use of the Confirm XML is monitored using a confirmation indicator set by the sending business software application. This is accomplished by setting the Confirmation Indicator (CONFIRMATION) in the Control Area of the originating XML. The Confirmation Indicators are defined as follows:

- 0 = Do not send back a confirmation XML.
- 1 = Send back a confirmation XML only if an error has occurred.
- 2 = Send a confirmation XML regardless of whether an error has occurred.

Confirm XML:

1. Ensures that the message was received and understood within the application or component
2. Communicates error conditions at the application level (such as when a field is missing, a customer does not exist, and so on).
3. Communicates that a critical update was successful (or unsuccessful), such as an update to inventory, credit balance, or ledger balance.

## Process PO

A Purchase Order is an XML business document that an organization issues to request delivery of goods or services for specific dates and locations.

The Process PO XML is used to transmit a purchase order to a supplier’s order management application. The Process PO is the task of sending the electronic form of a purchase order document to a supplier. This is designed as an external purchase order.

Data such as name, currency, payment method, bill-to and ship-to addresses, and  payment terms, along with information relevant to the supplier, is transmitted to the supplier to engage an order for goods and services.

## Acknowledge PO

The Acknowledge PO XML is a document used to acknowledge receipt of the Purchase Order and to reflect any changes. This is designed as an external purchase order.

Commonly, the acknowledgment is generated by an order management application and transmitted to a purchasing or procurement application. Acknowledgement (when the entire document is accepted, rejected, or modified) represents feedback from the supplier concerning the original purchase order received.

## Load Receivable

Receivable is an XML business document that transmits data to create a receivable open item in Oracle receivables from the billing information generated in an order management application. Receivable may also update the general ledger, depending on the specific architecture of the accounting application.

The scope of receivables is to create an XML to recognize customer obligations. Specific transactions include:

- Sales Invoice
- Credit Memo
- Debit Memo
- Charge Back

Receivable XML may also be used for transactions that do not originate from an order management application.

The receivable application may be a direct sub-ledger of the general ledger. Updates to G/L balances occur using the receivable module; therefore, the receivable XML contains both receivable and general ledger transaction information.

The other environment may also exist when general ledger updates occur directly from the Order Management application. The reconciliation between the receivable and general ledger is a function within the financial applications rather than of the integration space. This model allows the G/L balances to be updated in either detail or summarized form.

The role of the receivable application includes functions such as:

- Allowing Cash Application
- Dunning
- Dispute management

Relevant data that identifies the receivable includes header information (totals, identifier), partner location and contacts, invoice payment terms, and tax information.

## Load Payable

Payables is an XML business document that transmits data from the purchasing information generated in a purchasing application to create a payable open item in a payables application. This may also update the general ledger, depending on the specific architecture of financial applications.

Payables indicates that the supplier's invoice is ready to be paid and has been approved before the information moves to the accounts payable application. An approved invoice is also known as a voucher. The application later defines invoices that are matched within the accounts payable application in a separate business service request.

Some financial applications have the general ledger and accounts payable databases tightly integrated where updates to the accounts payable application are automatically reflected in general ledger balances. Payables transmits all information needed for both the accounts payable and the general ledger. Other applications allow the general ledger balances to be updated separately from the accounts payable. In this case, the payables and journals XML accomplishes this scenario.

## Sync Customer

The purpose of the Sync Customer XML is to keep customer information that exists on separate databases synchronized. The Sync Customer XML allows the adding of new customers and the modification of previously established customers. These customer records are used to reference invoices in a billing system.

## Sync Supplier

The Sync Supplier XML facilitates keeping supplier information that exists on separate data bases synchronized. The Sync Supplier allows the adding of new suppliers and the modification of previously established suppliers.

This record consists of relevant data to a partner (supplier name and address, contact information associated with the partner, and references of supporting documents).

## Load LdgrBudget

The LDGRBUDGET is an XML business document that transmits budget amounts from all possible source applications throughout an enterprise to a general ledger or budget application. Usually, budget data is created using a spreadsheet by each organization in an enterprise. Once budget data is ready, LDGRBUDGET XML facilitates the transfer of budget amounts against a revenue or expense account codes to a general ledger application for controlling expenditures and creating variance and analysis reports.

## Get PO and Show PO

The Get PO is an XML business service request that enables a business application module to request information concerning a specific purchase order from Oracle Purchasing. The reply to this request is the Show PO XML business request. There are a variety of business applications in several environments that may use this capability. For example, an MRP application may use this capability to ask for information from Oracle Purchasing, or a Plant Data Collection application may also use it to request information from Oracle Purchasing. This XML does not usually cause updates to occur.

As a Get PO request is made, the Show PO XML supplies purchase order information to another business application module. This request is also used as push notification of an event. There are many possible business applications in several environments that use this capability. For example:

1. Oracle Purchasing could use this request to send information to a Plant Data Collection application.
2. Other modules such as MRP, Inventory, or Manufacturing could use this to obtain order information.
3. The PO application can notify the MRP/Inventory application when a vendor gives or changes a promise to deliver.

## Receive PO

The Receive PO is an XML business service request that supplies the information that the Oracle Purchasing module requires to assign a receipt posting tag to a purchase order. Oracle Purchasing uses the receiving and inspection information supplied by the Receive PO XML to ensure that the organization only accepts and pays for the items ordered, received, or inspected, depending on the identified matching rule. A variety of business applications in several scenarios may use this capability. For example:

- Receipt of information from other non-Oracle systems.
- Receipt of barcoded and other receiving information from scanners.
- Advance Shipment Notices (ASNs) sent from suppliers.

As the Receive PO XML supplies the information, Oracle's Receiving Open Interface maintains the integrity of the new data as well as the receipt data already in Oracle Purchasing.

## Get Prodorder

Prodorder is an XML business document that details the work order information. This document may be related to three business service requests: Get Prodorder XML, Show Prodorder XML and Receive Prodorder XML. This enables the organization to import discrete job and repetitive schedule information, discrete job operations, and material, resource and scheduling information from any source.

### Show Prodorder and Receive Prodorder

The Get ProdOrder XML enables a business application module to request specific work order information from another business application module. Sources may include handheld devices, other manufacturing execution systems, planning systems, and order entry systems. The reply to this is the SHOW PRODORDER XML. Information on the following items may be used to request a work order: the pre-determined bill of material structure, lot, or serial information about the item; the operation in the routing at which to change the work order; and the accounting information that can optionally accompany a change in the order.

The Show ProdOrder is an XML business document that supplies work order information to another business application module. The environment for this request can be within the enterprise or outside the enterprise. On the other hand, the Receive Prodorder XML is a business service request that supplies information which the ERP system requires to do receipt posting against a work order.

## Transfer Item

The Transfer Item XML is a business service request that enables organizations to do sub-inventory or direct inter-organization transfers in Oracle inventory. As a company defines multiple inventories, warehouses, and manufacturing facilities as distinct organizations, the Transfer Item XML allows for the efficient performance of transfer of one or more items in a single transaction. This also allows the transfer of partial quantities of the same item to different sub-inventories and locations.

## Get Issueinfo and Show_IssueInfo

IssueInfo is an XML business service request for information against an order, from the ERP system into a plant data collection system to confirm the item issue transaction. The environment for this integration is from plant data collection systems to the Oracle manufacturing modules. Information concerning the work order to which the material item is being issued, the lot or serial number for the work order item, and the operation in the routing of the production item is transmitted to request an item issue.

The purpose of the Show IssueInfo XML business service request is to supply to another system item issue information against an order to confirm the issue transaction against that order. Examples of order types would be a work order, a sales order, a service order, or a maintenance order. This XML may be used individually, or as part of a larger interface scenario. Oracle Manufacturing (including Inventory and MRP) - to interface with Oracle or non-oracle Order entry systems, work in process systems, and the plant maintenance systems using this IssueInfo XML - allows for the issuance against an order.

## Confirm Issue

Issue is an XML business service request used to notify Oracle Manufacturing of the issue of required material to a work order for making a product. This XML is also used to notify Oracle Manufacturing of the return of material from a work order or job order back into inventory. The business environments most likely to require this capability are any type of manufacturing scenario.

The Issue XML communicates what item is being issued, where it is being issued from, which processing operation it is being issued to, what quantity was issued, and at what time this event occurred. In the case of a return, this communicates what item is being returned, which processing operation it is being returned from, which sub-inventory location it is being returned to, the quantity being returned, and the time at which this event occurred. This XML commonly causes updates to occur.

## Issue MiscItem and Receive MiscItem

The MiscItem XML is another business service request that reflects an unplanned issue or receipt of an item to or from a miscellaneous location. Possible reasons for issuance include somebody broke the material, or the material is defective and needs replacing, or the material is used up and needs replenishment. Miscellaneous transactions may be issued from a plant data collection system or an inventory system to Order Management or Manufacturing modules. The MiscItem XML may be used to issue materials to groups that are not inventory, receiving or work in process such as research and development group. This can also assist in the issuance of items to individuals or projects or to issue damaged items to expense accounts such as scrap. The MiscItem XML may also be used to notify a corresponding business application to reflect an unplanned receipt of an item to a miscellaneous location. This is primarily designed for supplies items or MRO items.

## Get Countinfo and Show Countinfo

Countinfo is an XML business service request to enable a business application to request and show inventory on-hand quantities from or to an Oracle Inventory system. The Countinfo XML enables a business application to request on-hand quantity information from an ERP system. This may also be used to show inventory count information to enable Oracle Inventory to send inventory count information to a PDC or other application system. This request may be used as a response to a get countinfo request or as a push notification of an event. There are many possible business applications in several environments that may use this capability.

## Update Invencount

Invencount is an XML business service request that transmits an inventory count to Oracle Inventory from the actual physical inventory location. This count may be a cycle count or a physical count. This XML may also apply to planned or unplanned inventory counts. The Invencount XML allows the user to transmit physical inventory count information that can e used to reconcile system-maintained item on-hand balances with the actual counts of inventory. Accurate system on-hand quantities are essential for managing supply and demand, maintaining high service levels, and planning production.

## Get Credit, Show Credit, and Update_credit

Credit is an XML business service request used to either Request, Show, or Update credit. This also includes Change Status request used to keep order, shipment and open items amounts current. This contains all of the information necessary to make a decision to give credit to a customer. The Request is made for the Order Management to request credit data for a customer from Oracle Receivables. This does not imply any update, it is only an inquiry function. The Show Credit XML is used as a response back to the order management application. The Update Credit XML may be used in both directions between the order management and the accounts receivable application. Its purpose is to keep order, shipment and open item amounts current. The Update Credit XML also transmits changes in the accounts receivable open item balances to the credit management function of the customer order management application.

## Change Status

The Change Status XML request is used to update the order management application with any changes in business status for a particular customer. The purpose of this request is to notify the customer order management application that the overall credit status of a customer has changed or status on specific order(s) are to be changed.

## Load Projacctg

Projacctg is an XML business service request that enables all relevant sub-systems (such as Accounts Payable for payment of materials used, Accounts Receivable for invoicing of billing projects, Budget for validation, Purchasing to report committed costs, Assets to record capitalized projects and Human Resources system to update employee information) that submit single sided transactions to send information to Oracle Project Accounting. Applications which produce double-sided transactions would use other XML business service requests to update the project accounting application. For example, order management or purchasing applications would use the Load Receivable or Load Payable XML. The Projacctg XML assists in providing project-oriented companies with a flexible approach to define and structure projects, tasks and budgets by which to monitor project status. Integration with other applications allows effective accounting for costs or to process revenue and invoices.

## Sync Projinfo

The purpose of the Sync Projinfo XML business service request is to enable all relevant sub-systems that submit transactions to Oracle Project Accounting to maintain valid values for the key project fields. The target applications for this update would include, but not necessarily be limited to: Accounts Payable, Accounts Receivable, Budget, Order Management, Purchasing, Time and Labor, Travel and Expense.

## Get Wipconfirm and Show Wipconfirm

Get WIP Confirm is an XML business service request that enables the requesting of data necessary to perform a confirmation of the movement of WIP (Work in Process). This does not commonly cause updates to occur. This template may be used individually, or as part of a larger interface scenario with a plant data collection system or a shop floor control system. Relevant data to get confirmation of the movement of WIP include: information concerning the specific work order or job order and the WIP operation or step in a routing from which the product is being completed into inventory or at which to return back into manufacturing, accounting information collected in the WIP Completion and Return transaction, information concerning the production order in the transaction as reference to an existing job order, and information concerning the resources associated with a particular WIP operation or sub-operation within a routing.

As a response to the Get WIP Confirm XML, the XML template Show WIP Confirm is used.

## Update Wipconfirm

Update WIP Confirm is an XML business service request that notifies Oracle Manufacturing of the completion of an end product in the production process and movement of that product to a finished goods inventory. This XML also notifies a Oracle Manufacturing of the return of end product from a finished goods inventory back into the production process. The business environments most likely to require this capability are any type of manufacturing scenario. This communicates which processing step the product is coming from, the quantity moving, the inventory organization or sub-inventory location it is moving to, and the time at which this event occurred. In the case of a return, the request communicates which inventory location the product is coming from, the quantity moving, the processing step it is moving to, and the time at which this event occurred.

## Get PickList

The Get PickList is an XML business service request that enables a request for the retrieval of a single Pick Slip from an ERP system. A pick slip or pick list is an internal shipping document pickers use to locate items to ship for an order. The reply to this request is the Show PickList. Individual lines from a Pick Slip are not selectable with this request. Only the complete document is selected and returned. Information relevant to the request include: a picking document that is generated in an ERP shipping module or Oracle Order Management.

## List PickList

List Picklist is another XML business service request that provides a list of Pick Slips from an ERP system to another application. This may be initiated in response to a GETLIST PICKLIST request or upon some business event. When a receiving application receives this XML, the information can be used as is or it may be used to initiate a selection of a specific Pick slip through the GET PICKLIST request. The processing is designed to provide multiple occurrences of summary data. This does not usually cause updates to occur.

The LIST XML is used for the receiver of the GETLIST XML to respond with the results of the search that was initiated by the GETLIST. Each of the records being returned is contained within an instance of the LIST_PICKLIST element underneath the DATAAREA Element.

The attributes associated with the LIST XML are as follows:

- rsstart attribute is a number that indicates the starting record for the section of the resulting set returned in the list message. This value should always match the rsstart value in the originating GetList XML.
- rscount attribute is a number that indicates the number of records returned in the message. The subsequent request for additional records should have a rsstart value of rscount + 1.
- rstotal attribute is a number that indicates the total number of records in the result set.
- rscomplete attribute is a Boolean that indicates that the list provided exhaust the possible values.
- rsref attribute is a string that represents the implementation-specific result set identifier for subsequent requests.

## Show PickList

The purpose of the SHOW PICKLIST XML is to show the details of an individual Pick Slip from an ERP system. This may be sent in response to a GET PICKLIST or it may be initiated upon some business event. This does not usually cause updates to occur. The picking document, the lines on a specific picking document, and details about a line item on a Pick Slip (such as date and time of loading and shipping) that are generated in an ERP shipping or order management system are some of the data that appears on the request.

## Update PickList

Update Picklist is an XML business service request that updates the details of an individual Picking List from a plant level to an ERP system. A pick slip or pick list is an internal shipping document pickers use to locate items to ship for an order. This usually causes updates to occur.

## Get Personnel and Show Personnel

Personnel is an XML business service request that may be used to either Get Personnel to request personnel data for a resource or Show Personnel to provide personnel data for a resource to a requesting business application. This facilitates integration between a human resource system to manufacturing systems such as shop floor and plant data collection. Personnel information such as employee name, cost center, division, employee category, status, job code, shift and wage group are required to manufacture, cost and schedule products.

## Update Persontime

The UPDATE PERSONTIME is an XML business service request that updates work time information for an employee from a data collection application to an ERP Human Resource application. This causes updates to occur. This may pass on data to update personnel information, employee category, employee status, overtime, and the quantity of employees' reporting hours, days, and so on.

## Sync Field

**Sync Field** is another XML business service request that enables the validation of data that exists on separate application's databases. This request can cause on-line validation to occur or may be a single tool for synchronizing data. In Oracle Financials, for example, this is very useful when a company engages in consolidation or inter-company transactions of which each balancing company resides in separate databases.

## Sync Personnel

The XML called Sync Personnel is a business service request that enables the synchronization of employee data that exists on separate databases between manufacturing and human resource applications. The Sync Personnel allows the adding of new employees and their relevant data as well as the modification of previously established employees. The Sync Personnel is used to facilitate the maintenance of human resource data in a manufacturing work force planning module, in Oracle this may be under Master Scheduling module. This enables the workforce planning module to use current personnel information when creating finite production schedules. This can also be used by Oracle Project Accounting or a work order management application to assign qualified personnel or to perform resource planning. Employee data such as employee name, category, job code, position, shift and competency are ut a few data that need to be synchronized between a manufacturing and a human resource application.

## Sync Wrkschdule

Sync Wrkschdule is an XML business service request that enables the synchronization of Work Schedule data that exists on separate databases. The Sync Wrkschdule allows the adding of new Work Schedules as well as the modification of previously established Work Schedules. This causes updates to occur and may be used as part of a large integration scenario or as a single tool for synchronizing data. This is designed primarily to enable synchronization of data in a Human Resource to Manufacturing Application integration scenario. Personnel work schedules detailed in a human resources system may be imported or exported to or from a manufacturing application such as Oracle Manufacturing.

## Sync Mfgtlcode

The **Sync Mfgtlcode** is an XML template that provides manufacturing codes that need to be captured with the labor hours in a time and labor reporting application. The Sync Mfgtlcode can also be used to provide other operational codes (such as medical codes) to a time and labor reporting application. The reason for associating these codes with labor hours is due to informational requirements of the manufacturing and/or financial systems.

The Sync Mfgtlcode may not be useful or necessary in integration scenarios in which one of the following is true:

1. The manufacturing codes observe complex hierarchical relationships or valid combinations. In this scenario, a plant data collection application or shop floor control application would probably be the labor reporting system,
2. Large volume repetitive manufacturing environments or process manufacturing in which none of the fields contained in the Sync Mfgtlcode need to be associated with labor hours by the manufacturing or financial applications.

## Sync COA

**COA** is an XML business service request that distributes general ledger chart of accounts (COA) code identifiers to other applications to store for validation purposes. This scenario assumes that the general ledger “owns” the chart of accounts definition and the instances of data within it. The sub ledger applications have several choices for validation of account numbers and other fields in the complete chart of accounts structure. One of these choices is to synchronize the chart of accounts structure and data from the general ledger application to all of the sub ledgers. The Sync COA allows for validation of account codes and account code combinations used in the subledger to record transactions and eventually post to the account balances in general ledger. The synchronization process can ensure that all of the components in the integration scenario have the same data necessary to communicate.

## Sync Exchngrate

The purpose of the **Sync Exchngrate** XML business service request is to enable the passing of updates of currency exchange rates to other applications that have exchange rate tables. In Oracle Financials for example, daily rates and period-end rates tables are maintained in Oracle GL. The sync Exchngrate XML facilitates the updating of exchange rates for other subsystems that require rates to convert or revalue foreign currency denominated transactions.

## Add Requisitn

**Add Requisitn** is an XML business service request that sends demand for goods or services to another business application for consideration of buying or in some way obtaining the requested items. In Oracle, this facilitates the sending of a purchase requisition to trigger the issuance of a purchase order for items requested. This XML usually causes updates to occur and may be used as part of a larger integration scenario or as a single tool for communicating demand. This request may originate from employees or requestor (using a self-service application), or may be triggered by Manufacturing modules such as MRP or Inventory when the reorder point reaches a level that signals the planner to issue a request for materials for production. Data such as supplier name, payment method and information that describes the requested item and its attributes including sub-components or sub-assemblies are sent to another business application (may be Oracle Purchasing) for validation and approval to purchase. The inclusion of a REQUISTNID (requisition ID) ties back a document to its origin, thus providing an identifier that enables drill back audit trail functionality.

## Change Requisitn

The **Change Requisitn** XML business service request communicates changes to an existing purchase requisition for goods or services. This change must refer to the original document and/or item requested. The change processing assumes replacement of fields sent, with the exception of REQUISTNID and REQLINENUM fields. If any of the Field Identifiers above require changing, that constitutes a cancellation of the request and/or the addition of another requisition. This may usually cause updates to occur and may be used as part of a larger integration scenario (between an Inventory or MRP system to a purchasing system) or as a single tool for communicating demand.

## Cancel Requisitn

The **Cancel Requisitn** XML business service request communicates from one business application to one or more other business applications that a previous requisition or requisition line item is no longer needed. This cancel must refer to the original document and/or item ordered. The user may either cancel the entire requisition or only specific lines on the requisition. To cancel the entire requisition, include only the REQUISITN Header information for the instance of the requisition you wish to cancel. To cancel a line or several lines, each line to be cancelled must be included. This XML commonly causes updates to occur and may be used as part of a larger integration scenario or as a single tool for communicating demand.

## Get Requisitn and Show Requisitn

An XML business service request called **Get Requisitn** enables a business application to request information concerning a specific requisition from another business application which is usually a purchasing system. The reply to this XML is the Show Requisitn XML. This Get Requisitn XML does not usually cause updates to occur. It may be used as part of a larger integration scenario or as a single tool for requesting information on existing demands for goods or services. Other users may need requisition information to validate statistics on growing demand for a specific item for planning purposes.

To send information relative to demand for goods or services to another business application or may be a notification vehicle initiated upon an event in a business application, the Show Requisitn XML business service request is used. The Show Requisitn XML usually does not cause updates to occur and may be used as part of a larger integration scenario or as a single tool for communicating demand. All data relevant to a specified purchase requisition (such as Supplier information, information describing the requested items and its attributes, sub-components and sub-assemblies) is sent via the Show Requisitn XML.

## Getlist Requisitn

**Getlist Requisitn** is an XML business service request enabling a business application to request summary information for one or more requisitions from another business application. The Getlist Requisitn also enables the retrieval of information across several documents by using selection fields. An example of this could be requesting all Requisition Lines for a specific ITEM. This type of functionality is limited to the capabilities of the responding application and needs to be determined during the implementation project. This XML does not usually cause updates to occur. It may be used as part of a larger integration scenario (between a manufacturing application and a purchasing application) or as a single tool for requesting information on existing demands for goods or services.

Many of the Data Types and the Field Identifiers contained within this XML may be used to select required information. This is done by requesting specific Field Identifiers or by requesting an entire Data Type. If the Data Type is not using Field Identifiers to select information, but the data within a Data Type is requested to be returned, the Data Type is coded without any Field Identifiers. This signifies to the responding application that all of the data that corresponds to that Data Type is to be included in the response. It is also possible to request that a range of values be returned the first occurrence of the XML is treated as the from and the second optional occurrence is treated as the to value. The maxitems attribute defined below indicates the maximum number of entries to return in the List XML that is used in response to this GETLIST.

## List Requisitn

To send information relative to demand for goods and services to another business application the **List Requisitn** XML business service request may be used. This may be in response to a Getlist Requisitn request, or it may be a notification vehicle, initiated upon an event in a business application. The LIST verb describes the behavior of supplying one or several documents in a summary format to the requesting business application. These listings of information may be supplied for requisition documents, or requisition lines and/or requisition sub lines. This XML usually does not cause updates to occur and may be used as part of a larger integration scenario or as a single tool for communicating demand.

The List XML is used for the receiver of the Getlist XML to respond with the results of the search that was initiated by the Getlist. Each of the records being returned is contained within an instance of the LIST_REQUISITN element underneath the DATAAREA Element.

## Getlist PO

**Getlist PO** is an XML business service request that enables a business application to request information containing summary information for one or more Purchase Orders from another business application. The Getlist PO also enables the retrieval of information across several documents by using selection fields. An example of this could be requesting all purchase order lines for a specific item. This type of functionality is limited to the capabilities of the responding application and needs to be determined during the implementation project. This does not usually cause updates to occur. It may be used as part of a large integration scenario or as a single tool for requesting information on existing demands for goods or services. A Plant Data Collection application could use this XML to request information from Oracle Purchasing or a MRP, Inventory or Manufacturing business application could use this to obtain order information.

Many of the Data Types and the Field Identifiers contained within this XML may be used to select required information. This is done by requesting specific Field Identifiers or by requesting an entire Data Type. If the Data Type is not using Field Identifiers to select information, but the data within a Data Type is requested to be returned, the Data Type is coded without any Field Identifiers. This signifies to the responding application that all of the data that corresponds to that Data Type is to be included in the response. It is also possible to request that a range of values be returned the first occurrence of the XML Element is treated as the from and the second optional occurrence is treated as the to value. The maxitems attribute defined below indicates the maximum number of entries to return in the List XML that is used in response to this GETLIST.

## List PO

A **List PO** XML business service request is used to send information relative to demand for goods or services to another business application. This may be in response to a Getlist PO request, or it may be a notification vehicle, initiated upon an event in a business application. The LIST verb describes the behavior of supplying one or several documents in a summary format to the requesting business application. These listings of information may be supplied for Purchase Orders, PO Lines, or PO Sub-Lines. This usually does not cause updates to occur and may be used as part of a larger integration scenario or as a single tool for communicating demand. There are many possible business applications in several environments that may use this capability. For example, a purchasing application could use this XML to send information to a Plant Data Collection application or a MRP, Inventory or Manufacturing business application could use this to obtain order information. The environment for this XML can be within the enterprise or outside the enterprise.

## Add PO

The purpose of the XML called **Add PO** is to communicate from one business application to one or more other business applications that a Purchase Order has been added or needs to be added, depending on the business case. The environment for this XML can be within the enterprise or outside the enterprise. There are many possible business applications in several environments that may use this capability. One example of a business integration scenario where this XML could be useful is to communicate back to an Inventory system that a purchase order has been processed and added in the purchasing system. The planner or buyer may use the purchase order information (such as payment terms, miscellaneous charges, the description and price of items ordered, and the dates and quantities for delivery or shipment of ordered products) to analyze and schedule production.

## Change PO

The purpose of the **Change PO** XML is to request another business application module such as Oracle Purchasing to make changes to an existing Purchase Order. This change must refer to the original document and/or item ordered. The change processing assumes replacement of fields sent, with the exception of the POID and the POLINENUM fields. If any of the Field Identifiers require changing, that constitutes a cancellation of the request and/or the addition of another Purchase Order.

## Cancel PO

The **Cancel PO** is an XML business service request that allows a requestor or planner to communicate from one business application to one or more other business applications that a previous Purchase Order or Purchase Order line is no longer needed. This cancel must refer to the original document and/or item ordered. A planner or requestor may invoke this XML to change the entire purchase order or to change specific purchase order lines. To cancel the entire order, include only the Purchase Order Header information for the instance of the Purchase Order you wish to cancel. To cancel a line or several lines, each line to be cancelled must be included. This XML commonly causes updates to occur and may be used as part of a larger integration scenario or as a single tool for communicating demand. In Oracle Purchasing, a cancellation of a purchase order eventually cancels accounting entries related to the order.

## Update Delivery

Delivery of goods from a supplier business partner via the services of a transportation provider (carrier) is an important event in a manufacturing or a purchasing application. As goods are delivered, Purchasing controls the items ordered through receiving, inspection, transfer, and internal delivery. These features control the quantity, quality, and internal delivery of the items delivered. The **Update Delivery** is an XML business service request that enables the update of delivery information for goods or services to another business application module. This XML can be initiated by a business application based on some event and sent to one or more relevant business applications. Possible business applications in several environments that may use this capability include: a Purchasing application to notify an Accounts Payable application of a specific delivery enabling the Accounts Payable application to accurately calculate the amount it needs to pay a business partner, a PO application could use this XML to send information to a Plant Data Collection application, a Purchasing application could use this to notify a MRP, Inventory, or Manufacturing business application that a delivery has occurred and the goods are available for use or inspection, and so on.

## Update Inspection

An **Update Inspection** XML business service request supplies inspection information for goods or services to another business application module. This may is initiated by the sending system upon some event occurring. There are many possible business applications in several environments that may use this capability. Examples include: a PO application could use this to send information to a Plant Data Collection application, or vice versa; a MRP, Inventory, Purchasing or Manufacturing business application could use this to communicate inspection information; a Laboratory Information System could send quality information to an Inventory application; or a Quality Control application could send information to a MRP, Inventory, or Purchasing application. Matching of supplier invoices against a purchase order may be defined in Oracle Purchasing or Oracle Payables at a four-way level, that is price and quantity billed must match quantity received (or delivered), ordered and inspected. The information supplied by the Update Inspection XML supports the four-way matching in Oracle. The XML also describes reasons for quantities rejected in the INSPECTION Data Type.

## Sync Item

**Sync Item** is an XML document that supplies information for goods or services to another business application module. This XML may also be initiated by the sending system upon some event occurring. This XML is not for synchronizing ITEM quantities at each inventory location. The Sync Inventory XML is used for this purpose. There are many possible business applications in several environments that may use this capability. For example, a MRP, Inventory, or Manufacturing business application could use this to communicate item information. This XML can be used to synchronize items used in finished goods, raw materials, work-in-process or components in a bill of materials. Data such as attributes of additional units-of-measure for the item, attributes of cost or value of an item and the location the item may be kept are synchronized for a more efficient handling of the items.

## Sync Sitelevel

**Sync Sitelevel** is an XML business service request that enables a mechanism to ensure that the physical location identifiers (physical locations of organizations or the location codes and their meanings) are synchronized between the business applications that require this to communicate clearly. This is particularly critical when only the codes that identify locations are used. Without the meaning of the codes clearly communicated, the integration is not effective. This XML enables the SITELEVEL codes to be synchronized among business applications. This may also be initiated by the sending system upon some event occurring. There are many possible business applications in several environments that may use this capability. For example, a MRP, Inventory, or Manufacturing business application could use this to communicate SITELEVEL information. The environment for this XML can be within the enterprise or outside the enterprise.

## Get Prodavil

**Get Prodavil** is an XML business service request that enables requests of product availability data by an Order Management business application (such as Oracle Management) to an Available to Promise (ATP) or Production business application. The scenario is the Order Management application interacting with the Available to Promise or Production application in order to determine availability of a product for the customer. This scenario is commonly referred to as Make to Order or Build to Order. The response to this request is the Show Prodavil. In the case where Finished Goods Inventory resides with the Order Management business application, Order Management may look at its' own Finished Goods Inventory before asking the ATP for product availability. Otherwise, The Order Management application always looks at the Available to Promise. The customer implementation rules or the business applications' capabilities usually determine this processing. In the case where there is a Finished Goods Inventory application on both sides of the integration, it is assumed that the Inventory applications are synchronized. In the case where Order Management is using the Production or Manufacturing Inventory, it is possible that Order Management would look at that Inventory availability before asking the Available to Promise application for product availability. This case is not covered at this time.

This XML usually does not cause updates to occur and may be used as part of a larger integration scenario or as a single tool for requesting product available to promise information. The picture below visualizes one possible use of this XML.

## Show Prodavil

The purpose of the **Show Prodavil** XML business service request is to respond to a Get Prodavil request or to initiate the passing of product availability data from a Production or Available to Promise (ATP) business application to an Order Management business application. The scenario is the Order Management application interacting with the Available to Promise or Production application in order to determine availability of a product for the customer. This scenario is commonly referred to as Make to Order or Build to Order. This may or may not cause updates to occur and may be used as part of a larger integration scenario or as a single tool for requesting product available to promise information. Data necessary to communicate ATP such as the item, item description, available quantity, date and time of availability is passed from a production application to an order management application.

## Update ProductReq

**Update Productreq** is an XML business service request that enables a business application such as Order Management to reserve a particular quantity of goods or services for a specific date and time. The scenario is the Order Management application interacting with the Available to Promise or Production application in order to determine availability of a product for the customer. This scenario is commonly referred to as Make to Order or Build to Order. In Oracle Applications, the Available to Promise is set up in the Order Management system, triggered by data from the manufacturing systems. The Update Productreq XML accomplishes this task in a two step process within this one request:

1. The receiving business application checks to see if an item is available in sufficient quantity by a specific date and time.
2. The receiving business application then reserves that quantity of inventory for that specific date and time combination if the product is available.

If the product requested is not available, the responding application may send one of two responses: either to use a Confirm XML to confirm the denial of the request or use a Show Prodavil to communicate an alternative product availability. This may be ITEM, DATE, or QUANTITY, or a combination of these. This may also be accompanied with a message in the NOTES field Identifier stating that this is an alternative. If the product requested is available, the responding application sends a Confirm XML business service request to confirm the execution of the request.

## Create Prodorder

A **Create Prodorder** XML notifies a Manufacturing Application of the need to make a product or parts in a specific quantity, for a specific need by date. The business environments most likely to require this capability are an Engineer to Order or a Configure to Order manufacturing scenario. This XML business request communicates what the product configuration is and what choices have been made from the configuration. This commonly causes updates to occur. As an order for a specific product is received in Order Management, a production order is created and an available to promise is set. The environment for this XML can be within the enterprise or outside the enterprise.

## Cancel Prodreq

**Cancel Prodreq** is an XML business service request that communicates from one business application to one or more other business applications that a previously requested item is no longer required. This cancel must refer to the original item requested. To cancel the item(s), each item to be cancelled must be included. This XML commonly causes updates to occur. Information required to identify the previously ordered item and to reset the order information gets communicated using this XML.

## Sync Inventory

The purpose of the **Sync Inventory** XML is to enable the synchronization of Inventory data that exists on separate Item Master databases. This data is not the master data that describes the attributes of the item such as dimensions, weight, or unit of measure. This is data that describes the ITEM as it exists at a specific location. The primary focus of this XML is to synchronize the quantity of an item by stocking location. This may create new Inventory records. Its purpose is to either create or update existing Inventory records.

## Get BOM and Show BOM

**Get BOM** (Bills of Materials) is an XML business service request that enables an application to request specific Item Bill of Material information from another business application module. The response to the Get BOM is the Show BOM. There are many possible business applications in several environments that may use this capability. For example, an MRP, Inventory, or Manufacturing business application could use this to communicate Item Bill of Material information. The environment for this XML can be within the enterprise or outside the enterprise.

The **Show BOM** XML is used to supply Item Bill of Material information to another business application module. This XML may also be initiated by the sending system upon some event occurring. There are many possible business applications in several environments that may use this capability. For example, an MRP, Inventory, or Manufacturing business application could use this to communicate Item Bill of Material information. Data relevant to the business such as information describing the BOM structure and its contents, information describing the attributes of a specific item or option within a classification are supplied by using the Show BOM XML. An example of an option would be CD-ROM for a laptop computer. Then each of the types of CD-ROM's for the option would be a separate ITEM. An example of an option class would be memory for a laptop. The options could then be 12, 24, or 40 megabytes of RAM. Each of these options would then have separate ITEM identifiers for memory modules that makes up the appropriate amount of memory. For 40 megabytes of RAM, this could be two 16 megabyte memory modules and an 8 megabyte module, or one 16 megabyte and three 8 megabyte memory modules.

## Sync BOM

**Sync BOM** (Bill of Materials) is an XML business service request that communicates to a business application module or system the need to initiate the creation of a Bill of Material structure. This XML may be necessary to address the Make to Order, Assemble to Order, or Mixed Mode business ordering scenarios in a Order Management to Manufacturing application integration scenario. In Oracle for example, Oracle Manufacturing and Oracle Order Management use bills of material to store lists of items that are associated with a parent item and information about how each item is related to its parent. Oracle Manufacturing supports standard, model, option class, and planning bills of material. There are many possible business applications in several environments that may use this capability. For example, an MRP, Inventory, or Manufacturing business application could use this to communicate the requirement to synchronize a Bill of Material. The synchronization process can ensure that all of the components in the integration scenario have the same data necessary to communicate. The environment for this XML can be within the enterprise or outside the enterprise.

## Allocate Activity

**Allocate Activity** is an XML business service request that enables the update of ACTIVITY information from a production or manufacturing application to a costing application. This is necessary for applications that are based on a Dual Cycle Accounting model. This Dual Cycle Accounting model does not capture the details of the activities that caused entries to be made in the general ledger application, but instead captures them in a separate overall costing application. For Single Cycle accounting systems, the Journals XML is used to ensure that the costing information flows from the Manufacturing Application to the Financial Application. This XML commonly causes updates to occur and may be used as part of a large integration scenario or as a single tool for updating data.

## Load Matchdoc

When using Oracle Applications, the purchase order and invoice matching functionality exists in Oracle Accounts Payable. But for other application suites, the matching functionality is made in the purchasing application. The invoice matching process may include several document types, including the following: for a Two way match - Purchase Order and the Invoice; for a Three way match - Purchase Order, Invoice, and the Receipt; and for a Four way match – Purchase Order, Invoice, Receipt, and Inspection results. For the four way match, it is assumed that inspection results have been updated on the Purchase Order for visibility in matching. The **Load Matchdoc** is an XML that is used to keep invoice, purchase order, goods receipt note and inspection ticket information current.

The Load Matchdoc XML is for use both by the accounts payable application and the purchasing application in exchanging the transactions that are required to be matched. When matching takes place in the accounts payable application, the purchasing application must inform the accounts payable application of the purchasing transactions (purchase orders, goods receiving notes and inspection tickets) to which the invoice (in accounts payable) is to be matched. These integration scenarios have been developed for document matching to occur at the line level within the PO document and the Invoice document. This may be a one to one relationship, or it may be a many to one relationship from Invoice to PO or from the PO to the Invoice. Charges not associated with a specific Invoice line must be matched individually. When matching takes place in the purchasing application, the accounts payable application may have to inform the purchasing application of the supplier invoice to which purchasing transactions (purchase orders, goods receiving notes and inspection tickets) are to be matched if the invoice is initially entered into the accounts payable application. Note that in some situations, invoices are entered directly into the purchase order application or are created by the purchase order application when using evaluated receipt settlement (ERS) and in this instance, it is not necessary to perform the separate integration described under this XML.

## Update Matchok and Update Matchfail

After loading matching documents and matching functionality is executed in an accounts payable application, an **Update Matchok** XML or an **Update Matchfail** XML is used to send successful matching notification or failure notification to a purchasing application. The purpose of the Update Matchfail XML is to notify the purchasing application of a matching failure such as a tolerance failure. For example in Oracle Payables, a user can match payables invoices to purchase orders to ensure that only goods ordered, received and inspected are paid. Controls and tolerances are set to specify the range of variance allowable if the amounts or quantities on the invoice are greater than the amounts or quantities on the purchase order or receipt. Oracle Payables then creates distributions and checks that the match is within the defined tolerance. Oracle Payables may use the Update Matchok XML or the Update Matchfail XML to notify the purchasing application. A purchasing application may then use this data to close or adjust the purchase order records or inform the supplier of discrepancies accordingly.

## Load_PLInvoice and Load Payable

**Load PLInvoice** XML is a business document that transmits data to create an unapproved open item in either a payables application or a purchasing application. The scope of the Load PLInvoice (load purchase ledger invoice) indicates that the supplier's invoice has not yet been approved and the invoice is to be used as part of the invoice matching process. For invoices that are approved for payment, this is handled by a separate XML business service request called **Load Payable XML**. If the matching functionality exists in a purchasing application, the Load PLInvoice XML is used to transmit data into the purchasing application, and only after matching does the PO application use the Load Payable XML to post approved invoices into the payables application. On the other hand, if the invoice matching functionality exists in the accounts payable application, the invoice is entered into accounts payable using the Load PLInvoice XML, and purchasing publishes matching document information to which accounts payable subscribes.

## Get Matchdoc and Show Matchdoc

A **Get Matchdoc** XML business service request is used to enable both the accounts payable application and the purchasing application to request the transactions that are required to be matched or request information concerning matching. In both cases, the receiving application uses the **Show Matchdoc** XML to return the requested information. This XML does not usually cause updates to occur. In certain application suites, purchase order and invoice matching functionality exists in the purchasing application, while in other suites this functionality exists in the accounts payable application. If the invoice matching functionality exists in the purchasing application, the invoice is entered into accounts payable, purchasing requests invoice information using Get Matchdoc XML and accounts payable provides invoice information using a Show Matchdoc XML. If the invoice matching functionality exists in the accounts payable application, the invoice is entered into accounts payable, purchasing requests matching document information using Get Matchdoc XML and accounts payable provides matching document information using Show Matchdoc XML.

## Getlist Picklist

The purpose of the Getlist Picklist XML business service request is to enable a business application to request summary information for one or more Picking Slips from an ERP system. If a list of documents is requested, that list is used in order for a selection and GET request of a specific picking list to be made, if necessary. This XML does not usually cause updates to occur. For example, this XML may be used by a Plant Data Collection system to request for summary information for a Pick slip from an Order Management system. Requestor may give specific field identifiers or request an entire data type, depending on the need for information.

## Update Inventory

**Update Inventory** is an XML business service request that enables the update of Inventory data that exists on separate Item Master databases. This data is not the master data that describes the attributes of the item such as dimensions, weight, or unit of measure. This is data that describes the ITEM as it exists at a specific location. The Update Inventory XML assumes that one Inventory Application is the "owner" of the data. Either the owner of the data or the non-owner of the data may initiate this XML. The Inventory Application which is the primary owner of the data is determined by which one updates the financial records to be posted to the general ledger. The non-owner inventory does not update the financial records. To do so may result in duplicate postings or other errors in the financial record keeping. All Inventory events which may affect the financial records need to be sent to the owner inventory either through the Update Inventory XML or the Sync Inventory XML. This does not create new Inventory records. Its purpose is to update existing Inventory records.

## Getlist Countinfo and List Countinfo

A **Getlist Countinfo** XML enables a business application to request several occurrences of summary Inventory Count information from an ERP system. This may be used for cycle counting or for physical inventory counts. The response to this request is the **List Countinfo** XML. This may be used individually, or as part of a larger interface scenario such as a Plant Data Collection and an Inventory system. The effective date and time, document ID or document type and the item may be used to get count information. This XML enables range selections. This is accomplished by including two separate occurrences of a Field Identifier. The first occurrence is the "FROM" selection and the second, or duplicate occurrence of the Field Identifier is the "TO" in the range to be selected. If a second Field Identifier is not included, the selection continues until the data no longer applies to the selection or the MAXITEMS is reached. All of the inventory documents may be requested by sending a value of "ALL" in the DOCUMENTID field.

A **List Countinfo** XML enables an Inventory application system to send multiple occurrences of summary or detail inventory count information to a Plant Data Collection or other application system. Data such as item description, quantity and serial number are examples of data that may be returned in the List Countinfo XML response.

## Cancel Prodorder

Cancel Prodorder is an XML business service request that notifies a Manufacturing Application of the need to cancel a previous order to make a product in a specific quantity, for a specific need by date. This XML may be used to cancel an entire Production Order, or a specific line on the production order. A cancel must refer to the original document and/or item ordered. To cancel the entire order, include only the header information for the instance of the Production Order that needs to be to cancelled. To cancel a line or several lines, each line (as indicated on the predetermined Bill of Material structure) to be cancelled must be included in the request. An Order Management system may send this request to a manufacturing system (in Oracle this is done under the Master Scheduling module) to cancel previously recorded Work or Job Order. Optionally, a user may also include information concerning the operation in the routing at which to change the production order in the Work in Process module.

## Sync Salesorder

The Sync Salesorder is an XML that facilitates the synchronization of sales or customer order information kept on separate databases throughout an enterprise. The Sync Salesorder allows the adding of new sales orders and the modification of previously established sales orders.

## Add Salesorder

For organizations with automated sales force systems, customer orders may be communicated to an Order Management system using the Add Salesorder XML business document. The Add Salesorder XML communicates from one business application to one or more other business applications that a Sales Order has been added or needs to be added, depending on the business case. interface scenario. Data pertinent to an order such as customer information and address (including the Bill To and Ship To addresses), sales person information, payment terms, any miscellaneous charges, an accounting distribution, the item or product ordered along with the quantity, price and other descriptive information and the schedule of delivery may be communicated effectively using the Add Salesorder XML.

## Cancel Salesorder

In order to communicate from one business application to one or more other business applications that a previous Sales Order, line, or schedule is no longer needed, the **Cancel Salesorder** XML may be used. This cancel must refer to the original document, item, and schedule. To cancel the entire order, include only the Salesorder Header information for the instance of the Salesorder to be cancelled. To cancel sales order lines and/or salesorder schedules, each line or schedule to be cancelled must be included in the occurrence of the XML with the SOLINENUM and SOSLINENUM identifiers specified respectively. If a schedule is to be cancelled, the line that the schedule refers to must be included or the schedule can not be found. This XML may be used by a salesperson who needs to cancel a pre-recorded order in an Order Management system.

## Change Salesorder

A Change Salesorder XML is a business service request that is used to request that another business application component make changes to an existing Sales Order. This change must refer to the original document and/or item ordered. The change processing assumes replacement of fields sent, with the exception of the fields for the sales order ID, the sales order line number, and the sales order schedule line number. If any of these Field Identifiers require changing, that constitutes a cancellation of the request and/or the addition of another Sales Order. This commonly causes updates to occur. A salesperson may decide to make changes to the orders already recorded in an Order Management system. This allows for flexibility in a system.

## Get Salesorder and Show Salesorder

**Get Salesorder** is another XML business service request that enables a business application module to request information concerning a specific sales order from another business application. The reply to this XML is the Show Salesorder. There are several possible business applications in several environments that may use this capability. For example, a Sales Automation application may use this XML to ask for information from a Order Management application. Request may be made by specifying the sales order ID and may optionally request for shipment and line details.

The **Show Salesorder** XML supplies Sales Order Information to another business application module such as a sales automation system. This request may be used as a response to a Get Salesorder request or as a push notification of an event. This XML does not usually cause updates to occur. Examples of what may be returned or shown include customer information, address, sales person information, and if necessary information on each order line and schedule. A sales person may also use the data returned by a Show Salesorder XML to know the commission amount to be appropriately credited for the order, assuming that the commissions have already been calculated.

## Getlist Salesorder and List Salesorder

To enable a business application module to request information containing summary information for one or more sales orders from another business application, such as Oracle Order Management, the **Getlist Salesorder** XML request may be used. The response to this request is the List Salesorder. The Getlist Salesorder also enables the retrieval of information across several documents by using selection fields. An example of this could be requesting all sales order lines for a specific item. This type of functionality is limited to the capabilities of the responding application and needs to be determined during the implementation project. This does not usually cause updates to occur. It may be used as part of a large integration scenario or as a single tool for requesting information on existing demands for goods or services. For example, a Sales Automation application may use this XML to ask for information from an order management application.

A **List Salesorder** XML is used to proactively send a listing of summary information about sales orders to one or more other applications. This does not usually cause updates to occur. It may be used as part of a large integration scenario or as a single tool for sending information concerning existing demands for goods or services. For example, an order management application may use this to respond to a request for information from a Sales Automation application.

## Sync PO

**Sync PO** is an XML business document that facilitates keeping purchase order information synchronized on separate databases throughout an enterprise. The Sync PO XML allows the adding of new purchase orders and the modification of previously established purchase orders. There are many possible business applications in several environments that may use this capability. One example of a business integration scenario where this XML could be useful is between a purchasing system and a Material Resource Planning (MRP) system under a manufacturing scenario. The environment for this XML can be within the enterprise or outside the enterprise.

## Sync Routing

Routing is the process an order must take in order to produce the finished goods. The **Sync Routing** XML is used to communicate to a business application component or system the need to create a new Routing or to update an existing Routing structure. This XML may be necessary to address the Make to Order, Assemble to Order, and Finished Goods business ordering scenarios in a Logistics to Manufacturing application integration scenario. There are many possible business applications in several environments that may use this capability. For example, an MRP, Inventory, or Manufacturing business application could use this to communicate the requirement to synchronize the Routing necessary to build finished goods. This XML may be used individually, or as part of a larger interface scenario. Routings between a Configuration Management system and a manufacturing system or with a Finite Scheduling system are examples where the Sync Routing XML can be used. The environment for this XML can be within the enterprise or outside the enterprise.

## Get Routing and Show Routing

The purpose of the **Get Routing** XML is to communicate to a business application module or system a request for an existing routing structure to be returned in a Show Routing XML. This may be used individually, or as part of a larger interface scenario.

**Show Routing** is an XML business service request that communicates to a business application module or system the relevant information about a specific routing. This XML may be used individually, or as part of a larger interface scenario. A Show Routing may be used to request a routing to be sent between a Routing and Configuration Management (or Oracle Bill of Materials module) system and a Manufacturing Execution System (Oracle Work in Process). This same scenario could exist between a Routing and Configuration Management system and a Finite Scheduling system. Relevant data such as the series of operations that create the routing, description of a particular item within a routing structure, a grouping of operations for the routing as well as a sequencing of operations, relationships between operations, the people needed within an operation and the description of the step within an operation for a specific routing are examples of information that may be communicated.

## Getlist Routing and List Routing

A **Getlist Routing** XML is used to communicate to a business application component or module a request for a summary list of a routing structure or structures to be returned in a List Routing XML. This XML may be used individually, or as part of a larger interface scenario. Example could be a Getlist Routing to request a list of possible routings to be sent via a List Routing between a Configuration Management system and a Manufacturing Execution System. This same scenario could exist between a Configuration Management system and a Finite Scheduling system.

The purpose of the **List Routing** XML is to communicate one or more summary listings of routing information to another business application component. This may be the result of a Getlist request between a Bill of Material system and a Work in Process system or it may be initiated by some other business event.

## Getlist BOM and List BOM

**Getlist BOM** (Bill of Material) is an XML that enables an application or component to request a summary list of Bill of Material information from another business application or component. The response to the Getlist BOM is the List BOM. The Getlist BOM also enables the retrieval of information across several documents by using selection fields. An example of this could be requesting all Bills of Material for a specific ITEM. This type of functionality is limited to the capabilities of the responding application and needs to be determined during the implementation project. There are many possible business applications in several environments that may use this capability. For example, an MRP, Inventory, or Manufacturing business application could use this to communicate Item Bill of Material information.

The **List BOM** XML is used to communicate one or more summary listings of BOM information to another business application component. This may be the result of a Getlist BOM XML or it may be initiated by some other business event. This XML may return information that generally describes the Bill of Material Structure and its contents, the attributes of a specific item, the attributes of a specific OPTION for an ITEM and information that describes the class of OPTION for a particular Product or Item.

## Get Item and Show Item

A **Get Item** XML enables a business application module to request information concerning a specific ITEM from another business application. The reply to this XML is the Show Item XML. There are many possible business applications in several environments that may use this capability. For example, an MRP, Inventory, or Manufacturing business application could use this to request item information. With Oracle applications, the Master Scheduling/MRP system uses item information to do forecasting of resources, Purchasing uses item information as basis for material replenishment and Inventory maintains these item information in its Master Item table.

To supply ITEM information to another business application module the Show Item XML can be used. This request may be used as a response to a GET ITEM request or as the result of some other business event. This XML does not usually cause updates to occur. There are many possible business applications in several environments that may use this capability. For example, item information may be communicated to a Purchasing Application from the Master Files kept in a MRP or Inventory system. The environment for this XML can be within the enterprise or outside the enterprise. Item information may include: general description of an item and its attributes, attributes of additional units-of-measure for the Item, attributes of cost or value for the Item and the location the item may be kept and the attributes of that location in relation to the item.

## Getlist Item and List Item

Item is anything made, purchased, or sold including components, sub-assemblies, finished goods or supplies. A **Getlist Item** XML business document enables a business application module to request summary information concerning an item or items from another business application. For example, an MRP, Inventory, or Work in Process application could use this to request item information from a purchasing application. This type of functionality is limited to the capabilities of the responding application and needs to be determined during an implementation project. Items are defined under Oracle Inventory and item information is used by other modules such as MRP, Inventory and Work in Process. The response to this request is the List Item XML. This does not usually cause updates to occur. The environment for this XML can be within the enterprise or outside the enterprise.

The List Item XML enables a business application module to respond to a Getlist Item request or to proactively send a listing of summary information about items to one or more other applications. Attributes of an item such as the description of the packing material to be used to package the item at its stocking unit of measure, the physical characteristics of the item, the shipping material to be used to ship the item and other attributes that generally describe the item defined in Oracle Inventory, may be listed for use by another application.

## Getlist Prodorder and List Prodorder

Prodorder is an XML business document that details the manufacture of a specific quantity of assembly, using specific materials and resources, in a limited time. For other applications, this may be termed as Work Order, Discrete Job or Assembly Order. With the **Getlist Prodorder** XML, a business software component such as a Receiving or Plant Data Collection system is able to request summary production order information from a Production system or Work in Process application. This XML also enables the retrieval of information across several documents by using selection fields. An example of this could be requesting all production order lines for a specific item.

The purpose of the **List Prodorder** XML request is to enable a business software component to respond to a request or to proactively send a listing of summary information about production orders to another business software component. This XML does not usually cause updates to occur. It may be used as part of a large integration scenario or as a single tool for requesting information on existing demands for goods or services. Information on individual items on the pre-determined Bill of Material structure, lot or serial number information about the final assembly defined in the production order, and the accounting information that can optionally accompany the production order may be listed.

## Sync Prodorder

The purpose of the **Sync Prodorder** XML request is to notify a manufacturing application of a change in the need to make a product. Such changes can in quantity or in need by date. The business environments most likely to require this capability are an engineer to order or a configure to order manufacturing scenario. This XML can communicate a new revision or change in configuration of the product being made.

## Sync Engchgordr

Engineering Change Orders (ECOs) enable control of new item revisions and bill of material changes. With one ECO, several bill of material changes can be grouped that affect one or more bills. Using Oracle Engineering, ECOs can be defined for all types of items and bills, including manufacturing and engineering items, bills and their components, planning, model, option class, and standard items, primary and alternate bills of material.

**Sync Engchordr** is an XML business service request that communicates to a business application module or system the need to initiate the creation of an Engineering Change Order. This XML may be necessary to address the make to order, assemble to order, or mixed mode business ordering scenarios in an order management to manufacturing application integration scenario. There are many possible business applications in several environments that may use this capability. For example, a PDM, MRP, inventory, or manufacturing business application could use this to communicate the requirement to synchronize an Engineering Change Order.

## Get Engchgordr and Show Engchordr

Engineering Change Orders (ECOs) enable control of new item revisions and bill of material changes. With one ECO, several bill of material changes can be grouped that affect one or more bills. Using Oracle Engineering, ECOs can be defined for all types of items and bills, including manufacturing and engineering items, bills and their components, planning, model, option class, and standard items, primary and alternate bills of material.

**Get Engchordr** is an XML business service request that communicates to a business application module or system the need to produce a Show Engchordr XML business document for the Engineering Change Order specified in the message. This XML may be necessary to address the make to order, assemble to order, or mixed mode business ordering scenarios in an order management to manufacturing application integration scenario.

To communicate to a business application module or system the sending systems representation of a specified Engineering Change Order, the Show Engchordr XML is used. The engineering change order is an instruction from design engineering or integrated product team that the approved design has been agreed by all stakeholder departments and is the valid method of manufacture on a given date and model unit.

## Confirm Engchordr

Engineering Change Orders (ECOs) enable control of new item revisions and bill of material changes. With one ECO, several bill of material changes can be grouped that affect one or more bills. Using Oracle Engineering, ECOs can be defined for all types of items and bills, including: manufacturing and engineering items, bills and their components, planning, model, option class, and standard items, primary and alternate bills of material.

The purpose of the **Confirm Engchordr** XML is to communicate to a business application module or system that the synchronization of a specified engineering change order has been completed successfully. This XML may be necessary to address the make to order, assemble to order, or mixed mode business ordering scenarios in a order management to manufacturing application integration scenario.

## Sync Dspchlist

A dispatch list is a list of tasks to be done by a manufacturing execution system in order to fulfill production orders. After production orders are sent out, master scheduling creates the dispatch list. Several applications (such as finite scheduling and ERP) provide a dispatch list to a manufacturing system.

**Sync Dspchlist** is an XML business service request used to synchronize dispatch list (finite schedule) information. This synchronizes information about the entire WIP transaction, information concerning the specific WIP operation, or step in a routing and information concerning the resources associated with a particular WIP operation. The synchronization process can ensure that all of the components in the integration scenario have the same data necessary to communicate.

## Get Dspchlist and Show Dspchlist

A dispatch list is a list of tasks to be completed by a manufacturing execution system in order to fulfill production orders. After production orders are sent out, master scheduling creates the dispatch list. Several applications (such as finite scheduling and ERP) provide a dispatch list to a manufacturing system.

The purpose of the Get Dspchplist XML business service request is to enable a business application module to request this information from another business application. The reply to this XML is SHOW. This XML does not usually cause updates to occur. This XML may be used individually, or as part of a larger integration scenario. For example, a manufacturing execution system or work in process module may get dispatch list information from a production planning module.

Show Dspchlist is an XML business document that communicates to a business application module, such as production planning, the sending systems representation of dispatch list (finite schedule) information. This XML may be used as a response to a Get Dspchlist request or as a push notification of an event. Information about the entire WIP transaction, specific WIP operation, or step in a routing and information concerning the resources associated with a particular WIP operation may be sent through the XML.

## Update Dspchlist

Update Dspchlist is an XML business service request that allows a user to update or make changes to an existing dispatch list (finite schedule) information. A dispatch list is a list of tasks to be completed by a manufacturing execution system in order to fulfill production orders. After production orders are sent out, master scheduling creates the dispatch list. Several applications (such as finite scheduling and ERP) provide a dispatch list to a manufacturing system. This XML causes updates to occur.

## Sync Maintorder

Sync Maintorder is an XML that ensures that all business software components in a specific integration instance have the current maintenance order information. This XML is commonly used to publish the need to create or update a Maintenance Order in a publish and subscribe integration environment. One possible scenario is the synchronization of Maintenance Order between field devices, service trucks, and so on, with a Computerized Maintenance Management System (CMMS). The synchronization process can ensure that all of the components in the integration scenario have the same data necessary to communicate. This data may include a description of the maintenance order, any accounting distribution information associated with the order, safety information related to work, location, and equipment, information on failure, cause and remedy, resources to perform the maintenance order, labor and tooling requirements to perform the operation, and information on the specific maintenance operation to be performed.

## Create Maintorder

The purpose of the Create Maintorder XML document is to publish to a business application component or system the need to create or update a maintenance order. This XML is commonly used to in a publish and subscribe integration environment. This XML may be used individually, or as part of a larger interface scenario. For example, creation of a maintenance order may be sent from a field device to a Computerized Maintenance Management System (CMMS).

## Update Maintorder

The purpose of the Update Maintorder XML document is to communicate any updates or changes necessary in an existing maintenance order. The environment for this XML can be within the enterprise or outside the enterprise. Maintenance orders detail the necessary maintenance operations for service equipment and tools used in the manufacturing scenario. Updates or changes may be made with the maintenance order information such as labor or craft resources, tool resources, operation, labor, and material resources or header information describing the maintenance order.

## Cancel Maintorder

Cancel Maintorder is an XML business service request that communicates from business application component such as a field device to one or more other business applications (that is, a computerized maintenance management system) that a previous maintenance order is no longer needed. This cancel must refer to the original document and/or maintenance order posted.

## Get Maintorder and Show Maintorder

The purpose of the Get Maintorder XML is to enable a business applications module to request order information from another business application. The response to this XML is SHOW. For example, a field device may want to get relevant information such as the resources to be used in the conduct of the maintenance operation, the operations that need to be performed under the order, and accounting distribution that may be necessary to relate the expenses of the maintenance to a particular cost center or department from a Computerized Maintenance Management system (CMMS).

In response, the CMMS creates a Show Maintorder XML to communicate data that meets the selection criteria posted under a Get Maintorder XML. The sending application sends what is requested in the Get Maintorder XML, if applicable, or alternatively, sends information that matches the data within the specified data type. A Show Maintorder XML communicates to a business application module or system the sending systems representation of maintenance order information.

## Getlist Maintorder List Maintorder

A Getlist Maintorder XML enables a business application module to request information containing summary information for one or more maintenance orders. The response to this request is the LIST. The environment for this XML can be within the enterprise or outside the enterprise. The Getlist Maintorder also enables the retrieval of information across several documents by using selection fields. An example of this could be requesting all RESOURCE data type occurrences for a specific OPERATION. This type of functionality is limited to the capabilities of the responding application and needs to be determined during the implementation project. This XML does not usually cause updates to occur.

A List Maintorder XML publishes one or more summary listings of maintenance order information to other business application component. This may be in response to a Getlist Maintorder request or to proactively publish a listing of summary maintenance order information for a business event. When a receiving application receives this XML, the information can be used as is or it may be used to initiate a selection of a specific maintenance order through the Get Maintorder XML request. The processing is designed to provide multiple occurrences of summary data.

## Sync UOMGroup

A UOMGROUP is a set of item-independent relationships that describe how an alternate units-of-measure is related to the stocking unit-of-measure or to other alternate units-of-measure. The environment for this XML can be within the enterprise or outside the enterprise. The purpose of the Sync UOMGROUP XML is to supply a set of unit-of-measure relationships to another business application module. This XML addresses the need for applications to exchange item-independent alternative UOM information beyond the stocking UOM. Item-independent implies that the materials management component or other subscribing applications manage the item's UOM as an entirely separate grouping that are later associated with one or more items.

This XML supports unit-of-measure (UOM) information flows associated with items being managed as inventory. The materials management, inventory control, warehousing and receiving, or shipping departments within an organization require the packaging relationship details between UOMs and the handling characteristics for each UOM to effectively manage the flow and storage of inventory. While the stocking level UOM provides for basic inventory accounting functionality, it may not be the most efficient, common, or natural UOM for manufacturing, purchasing, selling, or handling the item. The item may be packaged into standardized or supplier-specific bulk quantities for import or export, wholesale, or retail sales (for example, each, box, case, pallet, and so on).

## Get UOMGroup and Show UOMGroup

A UOMGROUP is a set of item-independent relationships that describe how an alternate units-of-measure is related to the stocking unit-of-measure or to other alternate units-of-measure.

The purpose of the Get UOMGROUP XML is to request an existing UOMGROUP structure or structures from a business application component or module to be returned in a Show UOMGROUP XML. There are many possible business applications in several environments that may use this capability. For example, an MRP, inventory, or manufacturing business application could use this to request alternate UOM information for one or more items.

The Show UOMGROUP XML transfers the UOM relationships independent of the item definition. This XML supplies Unit-of-Measure Group relationship information to another business application module. This request may be issued as a response to a Get UOMGROUP request or as the result of some other business event. This XML addresses the need for applications to exchange item-independent alternative UOM information beyond the stocking UOM. Item-independent implies that the materials management component or other subscribing applications manage the item's UOM as an entirely separate grouping that are later associated with one or more items. The environment for this XML can be within or outside the enterprise.

## Getlist UOMGroup and List UOMGroup

A UOMGROUP is a set of item-independent relationships that describe how an alternate units-of-measure is related to the stocking unit-of-measure or to other alternate units-of-measure.

The purpose of the Getlist UOMGROUP XML is to request a summary list of a UOMGRPHDR structure or structures from a business application component or module to be returned in a List UOMGROUP XML. The environment for this integration is application-to-application (A2A) within a single enterprise, or extended across enterprises, where there is a single master application that manages the assignment of all item identifiers. There are many possible business applications in several environments that may use this capability. For example, an MRP, inventory, or manufacturing business application could use this to request alternate UOM information for one or more items.

A List UOMGROUP XML supplies unit-of-measure group summary information to another business application module. This may be the result of a Getlist UOMGROUP request or some initiated by some other business event. When a receiving application receives this XML, the information can be used as is or it may be used to initiate a selection of a specific UOM group through the Get UOMGroup request. The processing is designed to provide multiple occurrences of summary data. The List XML is used for the receiver of the Getlist XML to respond with the results of the search that was initiated by the Getlist. Data such as those describing the packing material to be used to package the UOM, the inventory stocking unit of measure that can be tracked by an inventory control application, physical characteristics of the UOM (that is, height, volume, width), description of the shipping material to be used, are listed in summary by a List UOMGROUP XML to respond to a particular GetList XML.

## Sync Catalog

Sync Catalog is an XML document that communicates the need to initiate the creation of catalog information, as well as update existing catalog information, to a business application module or system. In communicating catalog information, the Sync Catalog may cause other information to be coordinated including item identifiers, specifications, pricing information agreed (that is, purchase agreements, price lists), availability and delivery information, and related items and accessories. This XML may be necessary to address the make to order, ordering scenarios in an Order Management to Manufacturing application integration scenario.

The existing definition does not attempt to model configuration rules, and is therefore not sufficient to model configure to order environments that require cross option validation. Assemble to order environments that can be modeled using a marketing bill of material are addressed. There are many possible business applications in several environments that may use this capability. For example, a Manufacturing, distributor or reseller business application could use this to communicate the requirement to synchronize a Catalog.

It may also be necessary to support Component Supplier Management (CSM) scenarios. In this scenario, a company provides a service of sourcing and codifying the products of many companies and publishing a consolidated catalog. A customer purchases the product from the Catalog provider. They have the capability to do comparison shopping from the catalog.

## Get Catalog and Show Catalog

The purpose of the GET CATALOG XML is to enable a business application module or system to request catalog information. In communicating Catalog information, the Get Catalog XML may cause other information to be coordinated including, but not limited to Item Identifiers, Specifications, Pricing Information agreed (that is, Purchase Agreements, Price Lists), Availability and Delivery Information and related items and accessories. This XML may be necessary to address the Make to Order or ordering scenarios in an Order Management to Manufacturing application integration scenario. There are many possible business applications in several environments that may use this capability. For example, a Manufacturing, distributor, or reseller business application could use this to get Catalog information.

It may also be necessary to support Component Supplier Management (CSM) scenarios. In this scenario, a company provides a service of sourcing and codifying the products of many companies and publishing a consolidated catalog. A customer purchases the product from the Catalog provider. They have the capability to do comparison shopping from the catalog.

A Show Catalog is an XML response to a Get Catalog request. The environment for this XML can be within the enterprise or outside the enterprise, including trading partners. The purpose of the Show Catalog XML is to supply a business application module or system with requested catalog information. Catalog information may include: fields that describe the catalog, identity of the publisher of the catalog, any category defined in the system to group specific items, description or listing of the items identified under the catalog, the specifications and specification value of a given item. The sending application sends what is requested in the Get Catalog XML, if applicable, or alternatively, sends information that matches the data in each of the data types.

## Sync Itemclass

The Item Classification XML may be used in both a business to business context and an application integration context. An example of this in an application integration context might be between a Product data management system and a procurement system to access outsourcing opportunities for items in a similar classification. Another example in a business to business context may be a component supplier management company letting a supplier know the marketing classification scheme it wishes to use in its catalog.

Sync Itemclass is an XML that communicates to a business application module or system the need to synchronize the classification and specification schemes. This may be necessary to address Manufacturers, Distributor Resellers business ordering scenarios in an Order Management to a Manufacturing application integration scenario. It may also be necessary to address Component Supplier Management scenarios.

There are many possible business applications in several environments that may use this capability. For example, a PDM, MRP, Inventory, or Manufacturing business application could use this to communicate the requirement to synchronize Item classification schemes. Examples may include, (but are not limited to): marketing classifications, production classifications, procurement classifications and shipping classifications.

## Get Itemclass and Show Itemclass

The Item Classification XML may be used in both a business to business context and an application integration context. An example of this in an application integration context might be between a Product data management system and a procurement system to access outsourcing opportunities for items in a similar classification. Another example in a business to business context may be a component supplier management company letting a supplier know the marketing classification scheme it wishes to use in its catalog.

Get Itemclass is an XML business document that enables a business application module or system to request information concerning classification and specification schemes. The environment for this XML can be within the enterprise or outside the enterprise, including trading partners. This XML may be necessary to address Manufacturers, Distributor Resellers business ordering scenarios in an Order Management to Manufacturing application integration scenario. It may also be necessary to address Component Supplier Management scenarios.

There are many possible business applications in several environments that may use this capability. For example, a Catalog Management System could use this to communicate the requirement to get Item classification schemes information from Product Data Management, Materials Resource Planning, Inventory, or Manufacturing business applications. The CLSSSCHMID Field Identifier is used as a selection field. This is the only Field Identifier value that should be sent in this GET request.

A Show Itemclass XML is used to supply a business application module or system with information concerning classification and specification schemes. This XML may be necessary to address Manufacturers, Distributor Resellers business ordering scenarios in a Order Management to Manufacturing application integration scenario. It may also be necessary to address Component Supplier Management scenarios. This XML returns information such as classification scheme description, data that describes an element or breakdown of the requested classification scheme, item category and item category assignments.

## Sync ITEMXREF

The Item cross-reference XML may be used in both a business to business context and an application integration context. An example of this in a business to business context might be a customer letting a supplier know the item numbering for a given European Article Number. An example of this in an application integration context might be between a Product Data Management system being the source system for Harmonized Schedule B numbers that are used by the transportation management system.

Sync ITEMXREF is an XML document that communicates to a business application module or system the need to synchronize an Item cross-reference. Cross-references may be to other item identifiers to the same form fit and function, as well as references to item identifiers of other items (form fit and function). For this XML, item relationships are used to refer to where the "to item" identifier, identifies a different form, fit and function to the "from item" identifier. It should be noted that the item identifier that is “Primary” in one system may be a secondary identifier in another system.

For example, in the Application Integration space, the manufacturing system may regard the “Item Number” as the primary identifier. The Order Management System may regard the Catalog number as the primary identifier. A company that manufactures hand held multi-meters may identify a given item in manufacturing with a 12 digit numeric code, 5432 123 12345. The marketing and sales department may refer to the same item by its catalog number of FL 30/4.

In the Business to Business case a supplier of hand held multi-meters may market their products through a catalog provider. The supplier has an item identifier with a corresponding party specific cross reference to the catalog providers identifier for the same item. The catalog provider has a item identifier for hand held multi-meters and a corresponding party specific cross reference to the suppliers item number. An example of this in a business to business context might be a customer letting a supplier know the valid substitutes that a supplier may supply to fulfill an order for a specific item number. An example of this in an application integration context might be between a Product Data Management (PDM) system and an Order Management system for accessories that may be offered to a customer with the sales of a major item. For example, if a designer of a video camera has designed it to work with accessories such as tripod, extended life battery pack, external microphone and head cleaner, the video may be designed to have the spares replaced by the consumer such as lens cover, strap and handle. The video camera may need the following consumable items on a recurring basis: video cassettes, batteries or lens cleaners. The relationship between these items and the video camera may need to be represented to the Customer in Web Based ordering or Customer Service Representative (CSR), in desk based order entry.

## Get ITEMXREF and Show ITEMXREF

The purpose of the Get ITEMXREF XML is to enable a business application module or system to request information concerning an Item cross-reference. Cross-references may be to other item identifiers to the same form fit and function, as well as references to item identifiers of other items (form fit and function). For this XML, item relationships are used to refer to where the “to item” identifier, identifies a different form, fit and function to the “from item” identifier. It should be noted that the item identifier that is “Primary” in one system may be a secondary identifier in another system. For example, in the Application Integration space, the manufacturing system may regard the “item Number” as the primary identifier. The Order Management System may regard the Catalog number as the primary identifier. A company that manufacture hand held multi-meters may identify a given item in manufacturing with a 12 digit numeric code, 5432 123

12345. The marketing and sales department may refer to the same item by its catalog number of FL 30/4.

This XML may be necessary to address Manufacturers, Distributor Resellers business ordering scenarios in a Order Management to Manufacturing application integration scenario. It may also be necessary to address Component Supplier Management scenarios. The Catalog Management System quotes a set of cross references that are not currently in the catalog. The cross-references may be either alternate identifiers of the products in the Catalog such as EAN Numbers and UPC Codes or they may be related items such as accessories, spares, or consumables. The Cross Reference document may flow from either the purchasing system of the selling system to let the Catalog Management System be aware of the identifiers used by all buying and selling parties.

A Show ITEMXREF XML supplies a business application module or system with information concerning an Item cross-reference. The XML may be used to relate item identifiers for item identifiers that identify different items (form fit and function). The Relationship types may also be universal. The sending application sends what is requested in the GET, if applicable, or alternatively, sends information that matches the data in the specified Data Type. The Show ITEMXREF XML allows the user to cross-refer either between classifications, or between items.

## Sync Pricelist

A Price List is a list containing the basic selling price per unit for a group of items, item categories, or services offered. Price lists are essential to ordering products because each item entered on an order must have a price. Most Order Management systems contain basic list information and one or more pricing lines, pricing attributes, qualifiers, and secondary price lists. Basic information for Oracle Order Management includes the price list name, effective dates, currency, pricing controls, rounding factor, and shipping defaults such as freight terms and freight carrier.

Sync Pricelist is an XML document that communicates to a business application module or system the need to synchronize product price list information. This allows the publication from the order management system of the Price List that accompanies the catalog. This XML may be necessary to address the Make to Order, Assemble to Order, or Mixed Mode business ordering scenarios in an Order Management to a Manufacturing application integration scenario.

There are many possible business applications in several environments that may use this capability. For example, a Manufacturing, distributor or reseller business application could use this to communicate the price change or request a price list. It may also be necessary to support Component Supplier Management (CSM) scenarios. In this scenario, a company provides a service of sourcing and codifying the products of many companies and publishing a consolidated catalog. A customer purchases the product from the Catalog provider. They have the capability to do comparison shopping from the catalog.

The primary workflow the Pricelist XML supports are to provide an instruction from a supplier to a customer that the supplier's list price for the suppliers product needs updating. It may also be an instruction from Marketing Systems to update order management systems. It may be an instruction from Order Management systems to Sales force automation systems to update Price Lists.

## Get Pricelist and Show Pricelist

A Price List is a list containing the basic selling price per unit for a group of items, item categories, or services offered. Price lists are essential to ordering products because each item entered on an order must have a price. Most Order Management systems contain basic list information and one or more pricing lines, pricing attributes, qualifiers, and secondary price lists. Basic information for Oracle Order Management includes the price list name, effective dates, currency, pricing controls, rounding factor, and shipping defaults such as freight terms and freight carrier.

The Get Pricelist XML business document enables a business application module or system to request information concerning new or existing product price lists. The environment for this XML can be within the enterprise or outside the enterprise, including trading partners. This XML may be necessary to address the Make to Order, Assemble to Order, or Mixed Mode business ordering scenarios in an Order Management to Manufacturing application integration scenario.

There are many possible business applications in several environments that may use this capability. For example, a Manufacturing, distributor or reseller business application could use this to request a price list. The Catalog Management System quotes a price list that does not exist in the Catalog Management System causing the Catalog Management System to request the price list. It may also be necessary to support Component Supplier Management (CSM) scenarios. In this scenario, a company provides a service of sourcing and codifying the products of many companies and publishing a consolidated catalog. A customer purchases the product from the Catalog provider. They have the capability to do comparison shopping from the catalog. Selection of price list to be returned may be made by providing a Pricelist ID using the Get Pricelist XML.

The purpose of the Show Pricelist XML is to supply a business application module or system with information concerning new or existing product price lists. This allows for the publication from the order management system of the Price List that accompanies the catalog.

The following types of data may be returned in a Show Pricelist XML:

- Price List Lines—the list of items or commodities and a base price for the items
- List price breaks—the prices and modifiers to the price for buying a given quantity or value of an item or item category on a price list line
- List Price Breaks—the prices and modifiers to the price for buying a given value of any product
- Price List Qualifiers—qualifies the Price Lists that may be used to price an item or item classification, in a given catalog, on a given date or for a given customer.

## Sync Itemspecs

The Item Specification XML may be used in both a business to business context and an application integration context. An example of this in an application integration context might be between a Product Data Management (PDM) system and a Procurement system to access outsourcing opportunities for items with similar specifications.

Sync ITEMSPECS is an XML document that communicates to a business application module or system the need to synchronize the specification of items within a catalog. The Item specification describes items in a catalog. The environment for this XML can be within the enterprise or outside the enterprise, including trading partners.

This XML may be necessary to address Manufacturers, Distributor Resellers business ordering scenarios in a Order Management to Manufacturing application integration scenario. It may also be necessary to address Component Supplier Management scenarios. There are many possible business applications in several environments that may use this capability. For example, a Product Data Management, Material Resource Planning, Inventory, or Manufacturing business applications could use this to communicate the requirement to synchronize Item specifications. Examples may include, (but are not limited to): Engineering systems communicating specification information to a purchasing system or a Supplier to a CSM company letting the CSM know the specifications of the items in the supplier's catalog.

## Get Itemspecs and Show Itemspecs

The purpose of the Get ITEMSPECS XML is to enable a business application module or system to request information concerning the specification of items within a catalog. The Item specification describes items in a catalog. The environment for this XML can be within the enterprise or outside the enterprise, including trading partners. This XML may be necessary to address Manufacturers, Distributor Resellers business ordering scenarios in a Order Management to Manufacturing application integration scenario. It may also be necessary to address Component Supplier Management scenarios. An example scenario for the Get ITEMSPECS XML could be that the specification documents quoted in the catalog are not yet in the Catalog Management System causing the Catalog Management System to request the item specifications from the Product Data Management system (PDM).

Using a Show ITEMSPECS XML, the PDM system publishes a set of Specifications for the items in the catalog. The purpose of the Show ITEMSPECS XML is to supply a business application module or system with information concerning the specification of items within a catalog. The Item specification describes items in a catalog. This XML may be necessary to address Manufacturers, Distributor Resellers business ordering scenarios in a Order Management to Manufacturing application integration scenario. It may also be necessary to address Component Supplier Management scenarios.

The sending application sends what is requested in the GET, if applicable, or alternatively, sends information that matches the data in the following data types:

- Feature Value information - defines the valid list of values for a given feature. An example of a feature value for a feature might be, operating voltage 110/240V AC.
- Feature Value Assignment information - describes the assignment of specifications or the values of specifications to a given item, or item category. An example of Feature Value Assignment information might be that a FL 856 hand held multi-meter may have an operating voltage of 110/240V AC.

## Sync RFQ

Request for Quotation (RFGQ) is a document used to solicit supplier quotations for goods or services needed. Possible sources of RFQs would be Purchasing, Master Scheduling or MRP, Inventory, or Work in Process systems. Another scenario of which the Sync RFQ XML refers to is the integration of a buyer’s business software components to supplier’s business software components.

Sync RFQ XML document ensures that all business software components in a specific integration instance have the current Request for Quotation information. Sync RFQ XML may be used in an integration scenario between a Buyer’s Purchasing system and a Seller’s Order Management system. The workflow sequence for custom, built to order involves more activity than that of finished goods. The buyer’s focus involves finding the best price for an item or quantity of items that are not currently available. A long-term business relationship is often involved leading to expectations that must be addressed within the quote. Customized products require a higher level of specification. Individual supplier inquiries may be answered formally for all participants to insure competitive parity. Conversely, buyers may have questions related to an individual quote that lead to a change. Sync RFQ XML is used when the RFQ data persists in multiple systems.

## Add RFQ

Request for Quotation (RFC) is a document used to solicit supplier quotations for goods or services needed. Possible sources of RFQs would be Purchasing, Master Scheduling or MRP, Inventory, or Work in Process systems. Another scenario of which the Add RFQ XML refers to is the integration of a buyer's business software components to supplier's business software components.

Add RFQ is an XML used to communicate from one business application to one or more other business applications that additional data related to a Request for Quotation has been added or needs to be added, depending on the business case. The environment for this XML can be within the enterprise or outside the enterprise. A Buyer's Purchasing System may use the Add RFQ XML to add a Request for Quotation to a Supplier's Order Management system.

The Add RFQ XML is used in pre-purchase activities for Quantities of Finished Items and Custom Build to Order Items. Add RFQ XML releases a request to one or more partners providing deadlines for response and item details. Add RFQ XML is used when the recipient takes ownership of the RFQ source data.

## Change RFQ

Request for Quotation (RFC) is a document used to solicit supplier quotations for goods or services needed. Possible sources of RFQs would be Purchasing, Master Scheduling or MRP, Inventory, or Work in Process systems. Another scenario of which the Change RFQ XML refers to is the integration of a buyer's business software components to supplier's business software components.

A Change RFQ XML requests that another business application component make changes to an existing Request for Quotation. The environment for this XML can be within the enterprise or outside the enterprise. A request is modified to include answers to questions or clarification of specifications using a Change RFQ XML.

## Cancel RFQ

Request for Quotation (RFC) is a document used to solicit supplier quotations for goods or services needed. Possible sources of RFQs would be Purchasing, Master Scheduling or MRP, Inventory, or Work in Process systems. Another scenario of which the Cancel RFQ XML refers to is the integration of a buyer’s business software components to supplier’s business software components.

The purpose of the Cancel RFQ XML is to publish to a business application or system the need to cancel an entire Request for Quotation or one or more of its line items. The environment for this XML can be within the enterprise or outside the enterprise. A Buyer organization’s Purchasing system may send a Cancel RFQ XML to a Supplier organization’s Order Management system to cancel an existing RFQ or lines of a RFQ.

## Respond RFQ

Request for Quotation (RFC) is a document used to solicit supplier quotations for goods or services needed. Possible sources of RFQs would be Purchasing, Master Scheduling or MRP, Inventory, or Work in Process systems. Another scenario of which the Respond RFQ XML refers to is the integration of a buyer’s business software components to supplier’s business software components.

A Respond RFQ XML communicates from one business application to one or more other business applications that additional data related to a Request for Quotation is available. The environment for this XML can be within the enterprise or outside the enterprise. The Respond RFQ XML is typically involved in a Custom Build to Order workflow. The workflow sequence for custom, built to order involves more activity than that of finished goods. The buyer’s focus involves finding the best price for an item or quantity of items that are not currently available. A long-term business relationship is often involved leading to expectations that must be addressed within the quote. Customized products require a higher level of specification. Individual supplier inquiries may be answered formally for all participants to insure competitive parity. Conversely, buyers may have questions related to an individual quotation that lead to a change. Suppliers respond with questions and/or clarification issues using a Respond RFQ XML.

## Getlist RFQ and List RFQ

Request for Quotation (RFC) is a document used to solicit supplier quotations for goods or services needed. Possible sources of RFQs would be Purchasing, Master Scheduling or MRP, Inventory, or Work in Process systems. Another scenario of which the Getlist RFQ XML refers to is the integration of a buyer’s business software components to supplier’s business software components.

Getlist RFQ is an XML that enables a business application module to request information containing summary information for one or more Request for Quotations. The response to this request is LIST RFQ. The environment for this XML can be within the enterprise or outside the enterprise. The Getlist RFQ also enables the retrieval of information across several documents by using selection fields. An example of this could be requesting all QUALFICATN Data Type occurrences for a specific RFQLINE. This type of functionality is limited to the capabilities of the responding application and needs to be determined during the implementation project. The typical scenario for use of this XML is that a Supplier or Buyer requests a list of RFQs.

The List is sent to the requestor by using a List RFQ XML. This XML publishes one or more summary listings of Request for Quotation information to other business application component. This may be in response to a Getlist RFQ request or to proactively publish a listing of summary Request for Quotation information for a business event. When a receiving application receives this XML, the information can be used as is or it may be used to initiate a selection of a specific Request for Quotation through the Get RFQ XML. The processing is designed to provide multiple occurrences of summary data.

## Get RFQ and Show RFQ

Request for Quotation (RFC) is a document used to solicit supplier quotations for goods or services needed. Possible sources of RFQs would be Purchasing, Master Scheduling or MRP, Inventory or Work in Process systems. Another scenario of which the Get RFQ and Show RFQ XML refer to is the integration of a buyer's business software components to supplier's business software components.

Get RFQ XML document enables a business applications module to request Request for Quotation (RFQ) information from another business application. The response to this XML is a Show RFQ. The environment for this XML can be within the enterprise or outside the enterprise. A Supplier's Order Management system sends a Get RFQ XML to a Buyer's Purchasing system to request RFQ information. It may also be that intermediaries are used to manage RFQ exchange activities between a Buyer and a Seller. The workflow sequence for scenarios using intermediaries involves additional activity. An intermediary may be independent or an extension of a large organization. Much of the additional activity follows from the aggregation provided by an intermediary. Multiple RFQs could reside with the intermediary. The Get RFQ XML may be used to select one of the items returned by a List RFQ XML for review. The full RFQ is sent by using a Show RFQ XML.

The Show RFQ XML communicates to a business application module or system the sending systems representation of Request for Quotation information. This request may be used as a response to a Get RFQ XML or as a push notification of an event.

## Sync Quote

A Quotation is a statement of the price, terms, and conditions of sale a supplier offers for an item or items. A quotation usually includes a detailed description (specifications) of goods or services the supplier offers. Suppliers consider quotations as an offer to sell when given in response to an inquiry. The integration scenario for the Sync Quote XML is the integration of a Buyer's business software components to Supplier's business software components.

A Sync Quote XML ensures that all business software components in a specific integration instance have the current Quotation information. This XML is commonly used to publish the need to create or update a Quotation in a publish and subscribe integration environment. The scenario of which the Sync Quote XML refers to is the integration of a Buyer's business software components to Supplier's business software components.

## Add Quote

A Quotation is a statement of the price, terms, and conditions of sale a supplier offers for an item or items. A quotation usually includes a detailed description (specifications) of goods or services the supplier offers. Suppliers consider quotations as an offer to sell when given in response to an inquiry. The integration scenario for the Add Quote XML is the integration of a Buyer's business software components to Supplier's business software components.

An Add Quote XML indicates a response to a supplier provided quotation or buyer question on a quotation. The purpose of the Add Quote XML is to communicate from one business application to one or more other business applications that additional data related to a quote has been added or needs to be added, depending on the business case. This XML commonly causes updates to occur. This XML may be used individually, or as part of a larger integration scenario. An example is the sending of an Add Quote XML from a Supplier's Order Management system to a Buyer's Purchasing system, or through an intermediary.

## Change Quote

A Quotation is a statement of the price, terms, and conditions of sale a supplier offers for an item or items. A quotation usually includes a detailed description (specifications) of goods or services the supplier offers. Suppliers consider quotations as an offer to sell when given in response to an inquiry. A quotation often represents a binding document that may change based on negotiations between the buyer or RFQ "owner" and the seller responding to the RFQ. The integration scenario for the Change Quote XML is the integration of a Buyer's business software components to Supplier's business software components.

The purpose of the Change Quote XML is to request that another business application component make changes to an existing Quotation. The environment for this XML can be within the enterprise or outside the enterprise. The Change Quote XML is sent by a Supplier's Order Management system to a Buyer's Purchasing system, or through an intermediary.

The Change Quote XML is used in pre-purchase activities for Quantities of Finished Items and Custom Build to Order Items. Using a Change Quote XML a Supplier may alter a quotation during negotiations in an effort to win the order.

## Cancel Quote

A Quotation is a statement of the price, terms, and conditions of sale a supplier offers for an item or items. A quotation usually includes a detailed description (specifications) of goods or services the supplier offers. Suppliers consider quotations as an offer to sell when given in response to an inquiry. A quotation often represents a binding document that may change based on negotiations between the buyer or RFQ "owner" and the seller responding to the RFQ. The integration scenario for the Cancel Quote XML is the integration of a Buyer's business software components to Supplier's business software components.

Cancel Quote XML is used to publish to a business application or system the need to cancel an entire quotation or one or more of its line items. The environment for this XML can be within the enterprise or outside the enterprise. This XML commonly causes updates to occur.

## Respond Quote

A Quotation is a statement of the price, terms, and conditions of sale a supplier offers for an item or items. A quotation usually includes a detailed description (specifications) of goods or services the supplier offers. Suppliers consider quotations as an offer to sell when given in response to an inquiry. The integration scenario for the Respond Quote XML is the integration of a Buyer's business software components to Supplier's business software components.

Respond Quote XML indicates a response to a supplier-provided quotation or buyer question on a quotation. The purpose of the Respond Quote XML is to communicate from one business application to one or more other business applications that an additional data related to a quotation has been added or needs to be added, depending on the business case. This XML commonly causes updates to occur.

## Getlist Quote and List Quote

A Quotation is a statement of the price, terms, and conditions of sale a supplier offers for an item or items. A quotation usually includes a detailed description (specifications) of goods or services the supplier offers. Suppliers consider quotations as an offer to sell when given in response to an inquiry. The integration scenario for the Getlist and List Quote XML is the integration of a Buyer's business software components to Supplier's business software components.

Getlist Quote XML enables a business application module to request information containing summary information for one or more quotations. The response to this request is LIST QUOTE. The environment for this XML can be within the enterprise or outside the enterprise. The Getlist Quote XML also enables the retrieval of information across several documents by using selection fields. An example of this could be requesting all SALESINFO Data Type occurrences for a specific QTELINE. This type of functionality is limited to the capabilities of the responding application and needs to be determined during an implementation project. This XML does not usually cause updates to occur. This XML may be used individually, or as part of a larger integration scenario. There are many possible business applications in several environments that may use this capability. An example is a Buyer's Purchasing system sending a Getlist Quote XML to a Supplier's Order Management system, or through an intermediary.

The purpose of the List Quote XML is to publish one or more summary listings of quotation information to other business application component. This may be in response to a Getlist Quote request or to proactively publish a listing of summary quotation information for a business event. The environment for this XML can be within the enterprise or outside the enterprise. When a receiving application receives this XML, the information can be used as is or it may be used to initiate a selection of a specific Quote through the GET QUOTE request. The processing is designed to provide multiple occurrences of summary data.

## Get Quote and Show Quote

A Quotation is a statement of the price, terms, and conditions of sale a supplier offers for an item or items. A quotation usually includes a detailed description (specifications) of goods or services the supplier offers. Suppliers consider quotations as an offer to sell when given in response to an inquiry. The integration scenario for the Get and Show Quote XML is the integration of a Buyer's business software components to Supplier's business software components.

Get Quote XML enables a business applications module to request this quotation information from another business application. The response to this XML is the SHOW QUOTE. The environment for this XML: can be within the enterprise or outside the enterprise. This XML does not usually cause updates to occur.

This XML may be used individually, or as part of a larger integration scenario. There are many possible business applications in several environments that may use this capability. An example is a Buyer’s Purchasing system sending a Get Quote XML to a Supplier’s Order Management system, or through an intermediary.

The purpose of the Show Quote XML is to communicate to a business application module or system the sending systems representation of quotation information. This request may be used as a response to a GET QUOTE request or as a push notification of an event. The environment for this XML can be within the enterprise or outside the enterprise.

## Show Shipment

Shipment is an XML business document that details the intent to transport a specific quantity of material goods from a supplier to a single customer business partner destination. The Shipment has been modeled after similar proprietary documents on popular business software packages (Oracle Applications' Delivery document, SAP's Delivery Note, and so on). The environment for this XML can be within the enterprise or outside the enterprise.

A Shipment is typically derived from the shipping schedule associated with a customer's purchase or sales order, once overall demand and various other business factors which prioritize the availability of the supplier's goods inventory have been evaluated. The Shipment document is designed to have a dynamic structure & content. Initial shipment planning information can be updated and significant detail (actual picked inventory attributes, ship unit packaging, and so on) may be added during the execution phase of the supplier's order fulfillment and shipping workflows. The final form of the Shipment document provides detail about the carrier and level of service used to transport the material, the exact quantity and attributes of the material shipped, and how that material is physically packaged and identified for transport.

To aid the customer's planning and receiving workflows, the supplier may transmit the final Shipment document to customer in advance so that they can prepare for carrier arrival and then efficiently accept and utilize the ordered material. In this use case, the Shipment document may function as a traditional Advance Ship Notice (ASN).

This XML may be used individually, or as part of a larger interface scenario. For example, a Shipping Management System uses a Show Shipment XML to show shipment information to a Buyer’s Purchasing System where the receipt of goods are recorded.

## Sync Planschd

PLANSCHD indicates a demand forecast sent from a customer to a supplier, or a supply schedule sent from a supplier to a customer. Sync Planschd XML communicates requirement information (part number, quantity, and so on) between a customer and their supplier on a regular basis, for example daily, weekly and so on, or for a user-defined time bucket type definition that is sent as part of this PLANSCHD.

Since collaboration between a customer and a supplier can potentially go through several rounds of negotiations, both parties can use the same XML to communicate their current demand or supply schedule in response to what they received from the other party. Either party can indicate detailed exception descriptions along with reason code and action code to the other party why the original demand or supply requirements need to be adjusted.

The Sync Planschd XML may be used individually, or as part of a larger interface scenario. An example would be the synchronization of plan schedules between a Buyer's Scheduling application and the Supplier's Release Management application. The Supplier Scheduling Application maintains schedules regarding planned demand, shipping and sequence of delivery. The Release management Application drives the MRP, the shipment and the sequence schedules in the Suppliers ERP application.

## Get Planschd and Show Planschd

PLANSCHD indicates a demand forecast sent from a customer to a supplier, or a supply schedule sent from a supplier to a customer. The Get Planschd XML enables business applications module to request plan schedule information from another business application. The response to this XML is Show PLANSCHD.

Since collaboration between a customer and a supplier can potentially go through several rounds of negotiations, both parties can use the same XML to communicate their current demand or supply schedule in response to what they received from the other party. Either party can indicate detailed exception descriptions along with reason code and action code to the other party why the original demand or supply requirements need to be adjusted.

This XML may be used individually, or as part of a larger interface scenario. An example would be the Supplier's Release Management application sending a Get Planschd XML to a Buyer's Scheduling application. It could be that the supplier Management application quotes a planning schedule that does not exist in the Demand Management application causing the Demand Management Application to request the planning schedule. This interface scenario assumes that the Supplier Scheduling Application maintains schedules regarding planned demand, shipping and sequence of delivery. The Release management Application drives the MRP, the shipment and the sequence schedules in the Suppliers ERP application.

Show Planschd XML communicates to a business application module or system the sending systems representation of plan schedule information. This request may be used as a response to a Get Planschd XML or as a push notification of an event. This XML is used to publish planning schedule from a Supplier Scheduling application.

## Sync Seqschd (Sequence Schedule)

Sync Seqschd XML enables the exchange of sequence schedule information authorizing a sequenced shipment of parts for specific trading partners and addresses. The environment for this XML can be within the enterprise or outside the enterprise. Commonly, the sequence schedule is generated by a work in process application and transmitted to an order or material planning application. This XML commonly causes updates to occur.

The Sync Seqschd XML may be used individually, or as part of a larger interface scenario. This event is the publication from the supplier scheduling application of a Shipment schedule to the buyer release management application. This interface scenario assumes that the Supplier Scheduling Application maintains schedules regarding planned demand, shipping and sequence of delivery. The Release management Application drives the MRP, the shipment and the sequence schedules in the Suppliers ERP application.

## Get Seqschd (Sequence Schedule) and Show Seqschd (Sequence Schedule)

A Get Seqschd XML enables a Release Management application to request sequence schedule information from another business application such as a Supplier Scheduling application. The response to this XML is Show SEQSCHD. This XML does not usually cause updates to occur.

Commonly, the sequence schedule is generated by a work in process application and transmitted to an order or material planning application. The environment for this XML can be within the enterprise or outside the enterprise. The SCHEDULEID Field Identifier is used as a selection field. This is the only Field Identifier value that is used to send a Get Seqschd XML.

A Show Seqschd XML communicates to a business application module or system the sending systems representation of sequence schedule information. This request may be used as a response to a Get Seqschd request or as a push notification of an event. Information detailing general description of partners, including addresses and contacts, sequence schedule date and time information, list of items ordered along with the quantity, delivery date and other schedule information is data that may be returned with a Show Seqschd XML. The information obtained may be used by a Demand Management application to do demand forecasting.

## Sync Shipschd (Shipment Schedule)

Sync Shipschd XML enables the exchange of shipment schedule information, authorizing a shipment quantity and date for specific trading partners and addresses. Commonly, the ship schedule is generated by a material planning application and transmitted to an order management application. Shipment schedules associated with a customer’s purchase or sales order is used to typically derive a shipment. This is completed once overall demand and various other business factors which prioritize the availability of the supplier's goods inventory have been evaluated.

The Sync Shipschd XML is used in an interface scenario between a Supplier Scheduling Application and a Release Management application. This interface scenario assumes that the Supplier Scheduling Application maintains schedules regarding planned demand, shipping and sequence of delivery. The Release management Application drives the MRP, the shipment and the sequence schedules in the Suppliers ERP application.

## Get Shipschd (Shipment Schedule) and Show Shipschd (Shipment Schedule)

The Get ShipSchd XML enables a Seller’s Release Management application to request shipment schedule information from a Buyer’s Supplier Scheduling application. This interface scenario assumes that the Supplier Scheduling Application maintains schedules regarding planned demand, shipping and sequence of delivery. The Release management Application drives the MRP, the shipment and the sequence schedules in the Suppliers ERP application. The response to this XML is the SHOW SHIPSCHD.

The Show Shipschd XML communicates to a business application module or system the sending systems representation of SHIPSCHD information. This request may be used as a response to a GET SHIPSCHD request or as a push notification of an event. Information on the partner including the location and contacts, the items ordered along with the quantity, delivery date and other schedule information and line item exceptions based on predefined business rules or contract agreements may be returned using a Show Shipschd XML.

## Process Invoice

Process Invoice is an XML that transmits an invoice from a supplier to a customer, thus involving an interface between a Supplier’s Accounts Receivable application with a Customer’s Accounts Payable application. Invoice in this interface scenario refers to either an item based customer invoice or as a general purpose customer credit or debit memo. A customer invoice is a document that is created in Receivables that lists amounts owed for the purchases of goods or services. This document also lists any tax, freight charges and payment terms. The Process Invoice XML transmits invoice information such as line items, charges, allowances and tax, partner information including location and contacts, invoice payment terms and information about a business partner document reference (that is, customer's purchase order, supplier's sales order) for the invoice as a whole or for a particular invoice line.

## Process WIPSPLIT

Process WIPSLIT (Work in Process Split) is an XML business service request that notifies a Manufacturing Application of the creation of multiple production lots from a single production lot of a product being made on a production order. The business environment most likely to require this capability is a lot-based discrete manufacturing scenario. The environment for this XML is normally within the enterprise. Process WIPSPLIT XML may be used in an integration that involves a manufacturing execution system (MES) and a central ERP system. The MES is the source of production events and activities and the ERP (with Order Management, Bill of Material, Cost Management and Inventory) is the source of related setup events which occur prior to production.

The Process WIPSPLIT XML communicates the originating lot, the resulting lots, their quantities, and the processing step at which this event occurred, along with the time at which this event occurred. This XML commonly causes updates to occur. Process WIPSPLIT XML may be used individually, or as part of a larger interface scenario. The process could start with the placement and synchronization of orders in the ERP system and the issuance of materials and charging of resources to the production order. The production order moves through the operations on the shop floor and throughout the production process, the order may experience a number of moves and splits or merges, and the creation of one or more bonus lots of material may possibly occur. The result of a WIP split may cause changes in the cost of the finished goods.

## Process WIPMERGE

Process WIPMERGE (Work in Process Merge) is an XML business service request that notifies a Manufacturing Application of the creation of a single production lot from multiple production lots of a product being made on a production order. The business environment most likely to require this capability is a lot-based discrete manufacturing scenario. The environment for this XML is normally within the enterprise. Process WIPMERGE XML may be used in an integration that involves a manufacturing execution system (MES) and a central ERP system. The MES is the source of production events and activities and the ERP (with Order Management, Bill of Material, Cost Management and Inventory) is the source of related setup events which occur prior to production.

The Process WIPMERGE XML communicates the originating lots, the resulting lot, lot quantities, and the processing step at which this event occurred, along with the time at which this event occurred. This XML commonly causes updates to occur. This XML may be used individually, or as part of a larger interface scenario. The process could start with the placement and synchronization of orders in the ERP system and the issuance of materials and charging of resources to the production order. The production order moves through the operations on the shop floor and throughout the production process, the order may experience a number of moves and splits or merges, and the creation of one or more bonus lots of material may possibly occur. The result of a WIP merge may cause changes in the cost of the finished goods.

## Process WIPRECOVER

Process WIPRECOVER is an XML business service request that notifies a Manufacturing Application of the creation of usable production materials from material previously considered unsuitable for production use. This is most often likely to represent a return to production of scrap material. The business environments most likely to require this capability are any type of manufacturing scenario (that is, discrete, process).

This XML communicates what is being recovered, the quantity being recovered, and the processing step at which the recovered material is to re-enter the production process, along with the time at which this event occurred. This XML commonly causes updates to occur. This XML may be used individually, or as part of a larger interface scenario. Process WIPRECOVER XML may be used in an integration that involves a manufacturing execution system (MES) and a central ERP system. The MES is the source of production events and activities and the ERP (with Order Management, Bill of Material, Cost Management and Inventory) is the source of related setup events which occur prior to production.

## Get WIPSTATUS and Show WIPSTATUS

Get WIPSTATUS is an XML business service request that enables a business application module to request information concerning the progress of a production order at a point in time from another business application. The business environments most likely to require this capability are any type of manufacturing scenario (that is, process, discrete) where business service requests for individual manufacturing transactions and events are not utilized. In a discrete manufacturing scenario, discrete job statuses describe the various stages in the life cycle of a job and control the activities that one can perform on it. Examples of Work in Process statuses are Complete, Released, Unreleased, On Hold, Cancelled or Close. The environment for this XML can be within the enterprise or outside the enterprise.

The Get WIPSTATUS XML communicates what quantities of end product reside at which processing steps along with the time this snapshot view was taken. The response to this XML is Show WIPSTATUS. This XML does not usually cause updates to occur. This XML may be used individually, or as part of a larger interface scenario. For example, a Work in Process system sends a Get WIPSTATUS XML to a Manufacturing Execution System to request the status or progress of a production order.

To respond to a Get WIPSTATUS XML, the Show WIPSTATUS XML is used. This Show WIPSTATUS XML notifies a Manufacturing Application of the progress of a production order at a point in time. The business environments most likely to require this capability are any type of manufacturing scenario (discrete, process) where business service requests for individual manufacturing transactions and events are not utilized. The environment for this XML can be within the enterprise or outside the enterprise. This XML communicates what quantities of end product reside at which processing steps along with the time this snapshot view was taken. This XML commonly causes updates to occur.

## Process WIPMOVE

Process WIPMOVE (Work in Process Move) is an XML business service request that notifies a Manufacturing Application of the progression through the production processing steps or operations of a product being made on a production order. The business environments most likely to require this capability are any type of manufacturing scenario (that is, discrete, process). The environment for this XML is normally within the enterprise. Process WIPMOVE XML may be used in an integration that involves a manufacturing execution system (MES) and a central ERP system. The MES is the source of production events and activities and the ERP (with Order Management, Bill of Material, Cost Management and Inventory) is the source of related setup events which occur prior to production.

The Process WIPMOVE XML communicates which processing step the product is coming from and which step it is being moved to, along with the quantity moving and the time this event occurred. This XML assumes that the applications involved in this business scenario have already synchronized the production item and its BOM or Routing information. This XML commonly causes updates to occur.

Process WIPMOVE XML may be used individually, or as part of a larger interface scenario. The process could start with the placement and synchronization of orders in the ERP system and the issuance of materials and charging of resources to the production order. The production order moves through the operations on the shop floor and throughout the production process, the order may experience a number of moves and splits or merges, and the creation of one or more bonus lots of material may possibly occur. In an environment such as Oracle Manufacturing, move transactions moves assemblies within an operation, such as from Queue to Run, or from operation to the next. Move transactions can charge resources and overheads.

## Allocate Resource

Allocate Resource is an XML that notifies a Manufacturing Application of the use of required labor or machine time on a production order making a product. The business environments most likely to require this capability are any type of manufacturing scenario (that is, discrete, process).

The Allocate Resource XML communicates what machine was utilized or which person performed the work and their labor skill class, along with the amount of time worked and at what time this event occurred. This XML commonly causes updates to occur. This XML may be used individually, or as part of a larger interface scenario. For example, interface between a Manufacturing Execution System and a Cost Management system to make updates to the cost of a product with the allocated resources. Information as to the resource transaction, work order the resource is to be charged against, resources to be charged, the production order against which the resources are to be charged, the lot and/or serial number information on the production items, the operation at which resources are to be charged, and the resources (worker, machine and tools) required to perform the operation assists in allocating the right resources for the order.

## Get Customer and Show Customer

Get Customer is an Way XML used to request customer information. This is likely between an Order Management system getting customer information from a customer management system. In Oracle Applications, customer records are maintained and managed in the Oracle Accounts Receivable system. Customer records include information on the customer (that is, customer name, payment method, partner type), customer's addresses and contacts. This XML does not cause updates to occur. An order in an Order Entry system can only be processed when customer records are in place and can be validated.

As a response to a GET Customer XML or a trigger by a business event, the Show Customer XML is used. The purpose of the Show Customer XML is to provide Customer information to a requesting business application, such as an Order Management system. This XML does not usually cause updates to occur.

## Get Supplier and Show Supplier

Get Supplier is an Way XML used to request supplier information. This is likely between an Order Management system getting supplier information from a supplier management system. In Oracle Applications, supplier records are maintained and managed in the Oracle Purchasing system. Supplier records include information on the supplier (that is, supplier ID, currency, payment method, partner type), supplier's addresses and contacts. This XML does not cause updates to occur.

The purpose of the Show Supplier XML is to provide Supplier information to a requesting business application, such as Order Management system. This XML may be in response to a Get Supplier XML, or it may be triggered by a business event. This XML does not usually cause updates to occur.

## Sync Ecatalog

Sync ECATALOG is an XML that synchronizes catalog information between two systems. The environment for this XML can be within the enterprise or outside the enterprise, including trading partners.

There are many possible business applications in several environments that may use this capability. Some examples of usage scenarios are a Manufacturer synchronizing catalogs with distributors, suppliers, or e-marketplaces and distributors, suppliers, or e-marketplaces synchronizing catalogs with buyers or other trading partners. It may also be necessary to support Component Supplier Management (CSM) scenarios. In this scenario, a company provides a service of sourcing and codifying the products of many companies and publishing a consolidated catalog.

The existing definition does not attempt to model configuration rules, and is therefore not sufficient to model configure to order environments that require cross option validation. Assemble to order environments that can be modeled using a marketing bill of material is addressed. The Catalog exchange scenario can be implemented either as a simple scenario using a single XML, or in the case of large catalogs involving complex pricing scenarios or partner specific details, as multiple XMLs.

## Get ECATALOG and Show ECATALOG

Get ECATALOG, is an XML that enables a business application module or system to request catalog information. The environment for this XML can be within the enterprise or outside the enterprise, including trading partners. The Catalog information that is requested by the Get CATALOG XML may include item identifiers, specifications, pricing information agreed on, price lists and availability and delivery information.

There are many possible business applications in several environments that may use this capability. Some examples of usage scenarios are Manufacturer exchanging catalogs with distributors, suppliers, or e-marketplaces and distributors, suppliers or e-marketplaces exchanging catalogs with buyers or other trading partners. It may also be necessary to support Component Supplier Management (CSM) scenarios. In this scenario, a company provides a service of sourcing and codifying the products of many companies and publishing a consolidated catalog. The Catalog exchange scenario can be implemented either as a simple scenario using a single XML, or in the case of large catalogs involving complex pricing scenarios or partner specific details, as multiple XMLs.

The purpose of the Show ECATALOG XML is to supply a business application module such as a Catalog Management System with requested catalog information. The environment for this XML can be within the enterprise or outside the enterprise, including trading partners. A Catalog Publisher should be able to publish catalog information to a customer or agent on information such as description of the product or service, party specific part identifiers, pricing information agreed on and other information depending on the needs of the subscriber. A Catalog Searcher should be able to request catalog information for a part or service identifier including description of product or service, part specific part numbers and other relevant information to the catalog.

## Sync Invoice

Sync Invoice is an XML that synchronizes or transmits an invoice from a supplier to a customer, thus involving an interface between a Supplier's Billing system with a Customer's Accounts Payable application. Invoice in this interface scenario refers to either an item-based customer invoice or as a general purpose customer credit or debit memo. A customer invoice is a document that is created in Receivables that lists amounts owed for the purchases of goods or services. This document also lists any tax, freight charges and payment terms. The Sync Invoice XML synchronizes invoice information such as line items, charges, allowances and tax, partner information including location and contacts, invoice payment terms and information about a business partner document reference (that is, customer's purchase order, supplier's sales order) for the invoice as a whole or for a particular invoice line.

## Get Invoice and Show Invoice

A Get Invoice XML enables a request of a customer invoice. This XML may be used as a request by a Customer to its Supplier. The interface would involve a Customer's Accounts Payable system requesting for invoice information from a Supplier's billing application. This allows the customer to get details of the invoice for processing and payment. A billing invoice may include information such as line items, charges, allowances and tax, partner information including location and contacts, invoice payment terms and information about a business partner document reference.

A response from the Supplier's billing system is to be carried through a Show Invoice XML. A Show Invoice XML transmits an invoice from a supplier to a customer. This XML may be used as a response to a Get Invoice request or as a push notification of an event.

## Get Consumption and Show Consumption

The most common use of the Get Consumptn XML is to request a buyer's usage information about an item or product for the supplier of such item or product. This XML does not create or update either buyer's or supplier's inventory records. The receiver of the request is responsible to make effective use of this information. The Get Consumption XML may be used for a supplier of goods to request from the buyer the consumption status of goods. This XML may also be used for a vendor to request from the retailer if retail sales of goods have been made and for inventory systems to request consumption status from plant data collection and warehouse management systems.

The most common use of the Show Consumtn XML is to share a buyer's usage information about an item or product with the supplier of such item or product. This XML does not create or update either buyer's or supplier's inventory records. The receiver of the request is responsible to make effective use of this information.

This is an outline of the business flow that these XML supports:

1. Overall purchase, replenishment or vendor managed inventory agreement is in place and/or a Get Consumptn message is sent by the supplier.
2. Show Consumptn Message is returned to the supplier, distributor or third party logistics provider, that material has been consumed. This is done in response to events such as these (and/or the Get message), depending on implementation context:
3. Material is replenished to line side at manufacturing facility.
4. Material is assembled into final product.
5. Material is purchased and removed from facility by customer.
6. Supplier, distributor, third party logistics provider replenishes material, using information provided in the Show Consumptn message, the demand and shipment forecasts, and the terms of the overall purchase or vendor managed inventory agreement.

## Create Requisitn

Create Requisitn is an XML request that notifies another business application of the need to order parts in a specific quantity, for a specific need by date. This XML usually causes updates to occur and may be used as part of a larger integration scenario or as a single tool for communicating demand. A Material Resource Planning (MRP) or an Inventory system may send a Create Requistn XML to a Purchasing System that handles purchase requisitions and orders maybe because reorder points indicate a need to request materials for production. Requisitions signify demand for goods and services to another business application for consideration of buying or in some way obtaining requested items. In Oracle, this facilitates the sending of a purchase requisition to trigger the issuance of a purchase order for items requested.

## Getlist LDGRACTUAL and List LDGRACTUAL

A **Getlist LDGRACTUAL** is an XML that requests information containing summary information for one or more ledgers. Many applications in the enterprise environment create data that causes changes in the account balances of a general ledger application (that is, Benefits, Costing, Human Resources and Payroll, Inventory, Manufacturing, Production, Treasury). The integration scenario for this XML is for enterprise applications business software components to obtain and receive accounting data from the general ledger business software component. The request or response style of messaging which the enterprise application sends a request message and expects to receive a response message back. The general ledger application receives the request message and produces the response message that is sent back to the enterprise application. The general ledger application uses information in the request message to know how to send the response message back to the enterprise application. This scenario assumes that the general ledger component “owns” the chart of accounts definition and the instances of data within it.

A List LDGRACTUAL XML is used to publish one or more summary listings of ledger information. This may be in response to a Getlist LDGRACTUAL request or to published proactively as a listing of summary ledger information for a business event. This ensures that subsidiary ledgers which keep details of the financial transactions are reflected correctly in summary in the general ledger application. Ledger actuals may also be used in the analysis of accounts

## Get LDGRACTUAL and Show LDGRACTUAL

Get LDGRACTUAL XML enables an enterprise application to request actual detailed accounting data from a general ledger system. General Ledger is a central repository of accounting data that is summarized into account balances for analysis and reporting in the books. Each subsidiary ledger usually sends journal data (usually in summary) to the general ledger. The general ledger system then posts this details to the appropriate accounts and books. The Get LDGRACTUAL XML is used to request specific ledger information.

The purpose of the Show LDGRACTUAL XML is to communicate to an enterprise application (subsidiary ledgers) the sending systems representation of ledger information specifically requested. This may be used as a to a Get LDGRACTUALS XML or as a push notification of an event (updated general ledger data). This XML may be used individually or as part of a larger interface scenario. This XML uses a Data Type called LDGRACTUAL, which represents any of the general ledger elements corresponding to the Account Structure hierarchy that can be queried.

## Acknowledge Delivery

A DELIVERY is an XML business document that details the receipt of goods by a customer or consignee destination from a supplier business partner using the services of a transportation provider (carrier). The DELIVERY document has been modeled for use in conjunction with the SHIPMENT XML when issued by the shipping business partner as an Advance Shipment Notification. The environment for this XML can be within the enterprise or outside the enterprise. DELIVERY was designed to support the physical process of receiving of goods from a carrier and is not biased toward receiving against a particular purchase, transfer or sales order reference document. This feature enables the receiving of goods simultaneously across multiple orders that were consolidated for shipping efficiency.

The Acknowledge Delivery XML may be used individually, or as part of a larger integration scenario. The ACKNOWLEDGE DELIVERY document may be used to notify the shipping business partner that the shipment has been received by the customer or consignee destination, and alert them to any discovered discrepancies. The acknowledgement may contain the full detail of the receipt as created by the receiving party or just the discrepancies and other exception conditions. This XML supports receipt acknowledgements at either the line item level and/or the ship unit level. Intermediate transportation or logistics providers or freight forwarding partners can use this document to acknowledge the receipt of entire shipping units without detailing the corresponding contents.

## Receive Delivery

A DELIVERY is an XML business document that details the receipt of goods by a customer or consignee destination from a supplier business partner using the services of a transportation provider (carrier). The DELIVERY document has been modeled for use in conjunction with the SHIPMENT XML when issued by the shipping business partner as an Advance Shipment Notification. The environment for this XML can be within the enterprise or outside the enterprise. DELIVERY was designed to support the physical process of receiving of goods from a carrier and is not biased toward receiving against a particular purchase, transfer or sales order reference document. This feature enables the receiving of goods simultaneously across multiple orders that were consolidated for shipping efficiency.

The Receive Delivery XML may be used individually, or as part of a larger integration scenario. The RECEIVE DELIVERY document may be used to update the receiver's internal receiving and order management business applications to indicate that the requested material has arrived, including any unexpected quantity, condition or other exception discrepancies. This XML supports receiving at either the line item level and/or the ship unit level. Intermediate transportation or logistics providers or freight forwarding partners can use this document to acknowledge the receipt of entire shipping units without detailing the corresponding contents.

## Get Delivery and Show Delivery

A DELIVERY is an XML business document that details the receipt of goods by a customer or consignee destination from a supplier business partner using the services of a transportation provider (carrier). The DELIVERY document has been modeled for use in conjunction with the SHIPMENT XML when issued by the shipping business partner as an Advance Shipment Notification. The environment for this XML can be within the enterprise or outside the enterprise. DELIVERY was designed to support the physical process of receiving of goods from a carrier and is not biased toward receiving against a particular purchase, transfer or sales order reference document. This feature enables the receiving of goods simultaneously across multiple orders that were consolidated for shipping efficiency.

The Get Delivery XML may be used individually, or as part of a larger integration scenario. For example, a Warehouse Management system may send a Get Delivery XML to an Order Management system. The GET DELIVERY XML may be used to request information about a specific expected (unreceived) or previously received goods delivery.

The Show Delivery is an XML document that may be used to obtain information about a specific expected (unreceived) or previously received goods delivery. The SHOW DELIVERY XML supports describing shipment content at either the line item level and/or the ship unit level. Intermediate transportation or logistics providers or freight forwarding partners can use this document to acknowledge the receipt of entire shipping units without detailing the corresponding contents. The Show Delivery XML may be issued in response to a Get Delivery request, or emitted asynchronously for notification upon some business event.

For expected deliveries, the SHOW DELIVERY document content may act as a receiving template or checklist to identify the quantity and shipping configuration of the expected goods. In this manner the SHOW DELIVERY XML may be considered logically equivalent to the Advance Ship Notice information in a SHOW SHIPMENT document. This similarity is by design, as the receiver’s SHOW DELIVERY may be directly derived from shipper’s SHOW SHIPMENT content after the receiver’s business logic has appropriately validated the Advance Ship Notification information.

## Getlist Delivery and List Delivery

A DELIVERY is an XML business document that details the receipt of goods by a customer or consignee destination from a supplier business partner using the services of a transportation provider (carrier). The DELIVERY document has been modeled for use in conjunction with the SHIPMENT XML when issued by the shipping business partner as an Advance Shipment Notification. The environment for this XML can be within the enterprise or outside the enterprise. DELIVERY was designed to support the physical process of receiving of goods from a carrier and is not biased toward receiving against a particular purchase, transfer or sales order reference document. This feature enables the receiving of goods simultaneously across multiple orders that were consolidated for shipping efficiency.

The GETLIST DELIVERY is an XML that is used to request information about a set of expected (unreceived) or previously received goods deliveries meeting certain selection criteria. The response to the GETLIST DELIVERY request is LIST DELIVERY XML. The GetList Delivery XML may be used individually, or as part of a larger integration scenario. For example, a Warehouse Management system may send a Getlist Delivery XML to an Order Management system to request information on previously received goods deliveries.

The LIST DELIVERY is an XML document used to obtain limited information listing about an expected (unreceived) or previously received goods deliveries that match certain selection criteria in a Getlist Delivery request. Additional information about a specific DELIVERY may be obtained through a Show Delivery XML by using the listing information to populate a GET DELIVERY request. The XML provides general information about the delivery receipt document, the partner including the address and contacts, information on the item inventory being delivered, information on the particular shipping unit being received, and specific quantity of goods contained within a shipping unit. The List Delivery XML returns specific data that matches the selection criteria provided in a Getlist Delivery XML.

# Samples

This section shows some simple examples of schemas and XML used in the adapter.

It includes the following topics:

- About the Sample Schemas
- About Sample XML to Test a Service

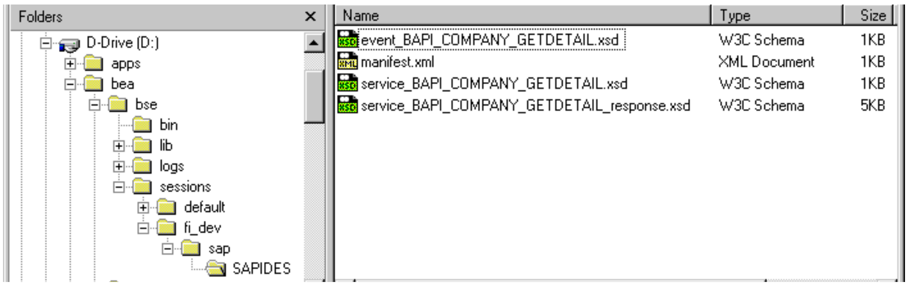
## About the Sample Schemas

This section describes the schemas created by the BEA Application Explorer.

- About the Directory Structure
- About the Manifest File
- About the Request Schema

### About the Directory Structure

The following figure illustrates a sample directory structure that BEA Application Explorer generated for the Oracle Applications connection named `SAPIDES` under the session named `fi_dev`.



| Name | Type | Size |
|---|---|---|
| event_BAPI_COMPANY_GETDETAIL.xsd | W3C Schema | 1KB |
| manifest.xml | XML Document | 1KB |
| service_BAPI_COMPANY_GETDETAIL.xsd | W3C Schema | 1KB |
| service_BAPI_COMPANY_GETDETAIL_response.xsd | W3C Schema | 5KB |

The generated metadata includes:

- a manifest file (`manifest.xml`)
- the service request schema (`service_BAPI_COMPANY_GETDETAIL.xsd`)
- the response schema (`service_BAPI_COMPANY_GETDETAIL_response.xsd`)
- the event schema (`event_BAPI_COMPANY_GETDETAIL.xsd`)

## About the Manifest File

Here is a simple example of a `manifest.xml` file generated for `BAPI_COMPANY_GETDETAIL`.

```
- <manifest>
 - <connection>
     <user>ib4</user>
     <password>july4</password>
     <ClientNumber>800</ClientNumber>
     <language>EN</language>
     <SystemNumber>00</SystemNumber>
     <hostName>esdsun2</hostName>
   </connection>
 - <schemaref name="BAPI_COMPANY_GETDETAIL">
     <request root="Company.GETDETAIL"
                          file="service_BAPI_COMPANY_GETDETAIL.xsd" />
     <response root="Company.GETDETAIL.Response"
                         file="service_BAPI_COMPANY_GETDETAIL_response.xsd" />
   </schemaref>
</manifest>
```

The adapter uses the information in the `manifest.xml` file to connect to the EIS from an application view. The manifest also points to the schemas for this adapter. These schemas define the adapter's interaction with the EIS.

You specify the location of the manifest and schemas when you configure the adapter during application view creation. To learn more about creating application views, see Chapter 3, "Defining Application Views for Oracle Applications."

The manifest file itself has the connection information, and points to the schemas for the adapter. In this simple example, there are only two schemas, request and response.

### About the Request Schema

This is a simple request schema generated for `BAPI_COMPANY_GETDETAIL`.

```
<xsd:schema targetNamespace="urn:sap-com:document:sap:business"
  <xsd:element name ="Company.GETDETAIL">
    <xsd:complexType>
      <xsd:all>
         <xsd:element name="COMPANYID" minOccurs="1">
           <xsd:simpleType>
             <xsd:restriction base="xsd:string">
               <xsd:maxLength value="6"/>
             </xsd:restriction>
           </xsd:simpleType>
         </xsd:element>
      </xsd:all>
      <xsd:attribute Name="COMPANYID">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:maxLength value="6"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

## About Sample XML to Test a Service

This is sample XML illustrating XML that tests a service. As with all these samples, this code applies to a particular situation. It is for illustrations purposes only.

- <eda>
- <request>
- <connection>
- <sp>
<proc>WIP_JOB_SCHEDULE_INTERFACE</proc>
<username>apps</username>
<password>apps</password>
- <data>
<LAST_UPDATE_DATE>2002-08-05</LAST_UPDATE_DATE>
<LAST_UPDATED_BY>1001611</LAST_UPDATED_BY>
<CREATION_DATE>2002-08-05</CREATION_DATE>
<CREATED_BY>1001611</CREATED_BY>
<LAST_UPDATE_LOGIN>NULL</LAST_UPDATE_LOGIN>
<REQUEST_ID />
<PROGRAM_ID />
<PROGRAM_APPLICATION_ID />
<PROGRAM_UPDATE_DATE />
<GROUP_ID>WIP_JOB_SCHEDULE_INTERFACE_S.NEXTVAL</GROUP_ID>
<SOURCE_CODE />
<SOURCE_LINE_ID />
<ORGANIZATION_ID>NULL</ORGANIZATION_ID>
<LOAD_TYPE>1</LOAD_TYPE>
<STATUS_TYPE>3</STATUS_TYPE>
<LAST_UNIT_COMPLETION_DATE />
<PROCESSING_WORK_DAYS />
<DAILY_PRODUCTION_RATE />
<LINE_ID />
<PRIMARY_ITEM_ID>155</PRIMARY_ITEM_ID>
<BOM_REFERENCE_ID />
<ROUTING_REFERENCE_ID />
<BOM_REVISION_DATE />
<ROUTING_REVISION_DATE />
<WIP_SUPPLY_TYPE>7</WIP_SUPPLY_TYPE>
<CLASS_CODE>Discrete</CLASS_CODE>
<LOT_NUMBER />
<JOB_NAME />
<DESCRIPTION />
<FIRM_PLANNED_FLAG />

```
<ALTERNATE_ROUTING_DESIGNATOR />
<ALTERNATE_BOM_DESIGNATOR />
<DEMAND_CLASS />
<START_QUANTITY>100</START_QUANTITY>
<WIP_ENTITY_ID />
<REPETITIVE_SCHEDULE_ID />
<ATTRIBUTE_CATEGORY />
<ATTRIBUTE1 />
<ATTRIBUTE2 />
<ATTRIBUTE3 />
<ATTRIBUTE4 />
<ATTRIBUTE5 />
<ATTRIBUTE6 />
<ATTRIBUTE7 />
<ATTRIBUTE8 />
<ATTRIBUTE9 />
<ATTRIBUTE10 />
<ATTRIBUTE11 />
<ATTRIBUTE12 />
<ATTRIBUTE13 />
<ATTRIBUTE14 />
<ATTRIBUTE15 />
<INTERFACE_ID />
<LAST_UPDATED_BY_NAME />
<CREATED_BY_NAME />
<PROCESS_PHASE>2</PROCESS_PHASE>
<PROCESS_STATUS>1</PROCESS_STATUS>
<ORGANIZATION_CODE>M1</ORGANIZATION_CODE>
<FIRST_UNIT_START_DATE>2002-08-05</FIRST_UNIT_START_DATE>
<FIRST_UNIT_COMPLETION_DATE>2002-08-05</FIRST_UNIT_COMPLETION_DATE>
<LAST_UNIT_START_DATE>2002-08-05</LAST_UNIT_START_DATE>
<SCHEDULING_METHOD />
<LINE_CODE />
<ROUTING_REVISION />
<BOM_REVISION />
<COMPLETION_SUBINVENTORY />
<COMPLETION_LOCATOR_ID />
<COMPLETION_LOCATOR_SEGMENTS />
```

```
<SCHEDULE_GROUP_ID />
<SCHEDULE_GROUP_NAME />
<BUILD_SEQUENCE />
<PROJECT_ID />
<TASK_ID />
<NET_QUANTITY>100</NET_QUANTITY>
<PROJECT_NUMBER />
<TASK_NUMBER />
<END_ITEM_UNIT_NUMBER />
<OVERCOMPLETION_TOLERANCE_TYPE />
<OVERCOMPLETION_TOLERANCE_VALUE />
<KANBAN_CARD_ID />
<PRIORITY>2</PRIORITY>
<DUE_DATE>2002-08-05</DUE_DATE>
<ALLOW_EXPLOSION>Y</ALLOW_EXPLOSION>
<HEADER_ID />
<DELIVERY_ID />
</data>
</sp>
</connection>
</request>
</eda>
```

# Index

## A

adapter
  benefits 1-11
adapters
  defined 1-4
Advance Queuing (AQ), defined 1-10
Advanced Queuing (AQ) system 1-10
Application Explorer
  about the BEA Application Explorer 2-2
  starting 2-4
application views
  adding events to 3-13
  adding services to 3-7
  defined 1-5
  events, adding 3-13
  final configuration tasks 3-19
  overview of defining 3-3
  preparing to define 3-2
  services, adding 3-7
  services, testing 3-21
  testing events manually 3-23
  testing events using a service 3-22
  testing services 3-21
AQ (Advanced Queuing) system 1-10
auditing events 3-20

## B

BEA WebLogic Adapter for SAP, overview of 1-9
benefits of adapter 1-11

## C

concurrent program
  determining request parameters 3-24
  request document 3-27
  submit request document 3-30
concurrent program agent, testing 3-23
concurrent programming, defined 1-10
concurrent programs 2-3
concurrent programs agent 3-10
control table
  described 4-5
control table, described 4-2
customer support contact information xv

## D

default session path, changing 2-4
defining schemas 2-2
Delete Keys 4-6
delete keys 3-15, 3-17, 4-3
determining concurrent program request parameters 3-24

## E

enterprise information systems, defined 1-4
event key
  defined 4-2
events
  about 2-3
  adding to application views 3-13

auditing 3-20
defined 1-4
event schemas 1-8
row tracking 4-1
testing 3-22
testing manually 3-23
testing using a service 3-22

## G

getting started 1-11

## I

interface tables 3-7
iProcurement Connector 1-10

## L

logging 3-20
logon parameters, specifying 2-5

## O

OAGIS (Open Applications Group Integration Specification) 1-10
Open Applications Group Integration Specification (OAGIS) 1-10
Oracle Advanced Queuing 3-9
Oracle Advanced Queuing (AQ) event 3-17
Oracle AQ 3-9
Oracle connections
creating a new connection 2-6
editing an existing connection 2-7
logon parameters 2-5
Oracle interface tables 3-13
Oracle JDBC driver 3-7, 3-8, 3-13
OracleAppsTable event delete keys 3-17
OracleAppsTable event parameter SQL Post Query 3-16
OracleAppsTables 3-13

## P

product support xvi

## R

related information xiv
request document 3-30
request document, for concurrent program 3-27
request schemas 1-8
response schemas 1-8
row removal event
configuring 4-3
row removal events 4-2
row tracking event
configuring 4-2
Row tracking events 4-1

## S

schemas
concurrent programs 2-3
defined 1-7
defining 2-2
event schemas 1-8
events 2-3
request schemas 1-8
response schemas 1-8
service requests 2-3
service responses 2-3
service requests 2-3
service responses 2-3
services
adding to application views 3-7
defined 1-4
request schemas 1-8
response schemas 1-8
testing 3-21
translation 1-8
session path, changing 2-4
source table

described 4-1
SQL Post Query 3-15, 3-16, 4-2
support xvi
supported SAP operations 1-10

## T

technical support xvi
testing the concurrent program agent 3-23
trigger-based events
configuring 4-5
described 4-3
triggers
described 4-3