Oracle® Retail WebTrack Operations Guide Release 12.0 May 2006

ORACLE

Contents

Preface

This operations guide serves as an Oracle Retail WebTrack reference to explain ‘backend’ processes. The guide is designed so that you can view and understand key system administered functions, the flow of data into and out of the application, and the application’s behind-the-scenes processing of data.

Audience

Anyone who has an interest in better understanding the inner workings of the Oracle Retail WebTrack system can find valuable information in this guide. There are three audiences in general for whom this guide is written:

- System analysts and system operation personnel:
  - who are looking for information about the application’s processes internally or in relation to the systems across the enterprise.
  - who operate Oracle Retail WebTrack on a regular basis.
- Integrators and implementation staff who have the overall responsibility for implementing Oracle Retail WebTrack into their enterprise.
- Business analysts who are looking for information about processes and interfaces to validate the support for business scenarios within Oracle Retail WebTrack and other systems across the enterprise.

Related Documents

This guide does not show you how to use the front-end of Oracle Retail WebTrack. Rather, the focus here is on how data is managed, how it flows, and how it is processed.

If you wish to find further information, see the following applicable Oracle Retail documents, as well as the Release Notes for WebTrack and for Design (not shown below):

- Oracle Retail WebTrack User Guide
- Oracle Retail WebTrack Online Help
- Oracle Retail WebTrack Configuration Guide
- Oracle Retail Design User Guide
- Oracle Retail Design Online Help
- Oracle Retail Design Configuration Guide
- Oracle Retail Integrator User Guide
- Oracle Retail Retail Server Installation Guide
- Oracle Retail Retail Server Data Model
- Oracle Retail Integrator User Guide and Online Help

Customer Support

- https://metalink.oracle.com

When contacting Customer Support, please provide:

- Product version and program/module name.
- Functional and technical description of the problem (include business impact).
- Detailed step-by-step instructions to recreate.
- Exact error message received.
- Screen shots of each step you take.

1
Introduction

Oracle Retail WebTrack is a web-based, collaborative critical path management solution which brings the members of a client's supply chain together to track critical, time-sensitive business activities and events.

In today's business climate, clients wish to take advantage of strategic opportunities. For example, they may wish to expand the ratio of import to domestic products, or shift to a more profitable private-label branding strategy. To accomplish these objectives, the application's ability to track and manage the development and global sourcing of goods becomes paramount to competitive success. Oracle Retail WebTrack is a critical path management tool that binds the client with its multitude of trading partners (such as third-party agents, manufacturing suppliers, raw materials vendors, and so on) to manage the complex process of bringing goods to market.

The system provides a solution to issues raised around communication methods, supplier reaction time, workload balancing, and data management. Enhanced visibility and improved communication with supply chain partners brings order and control to the event management process. Using Oracle Retail WebTrack to work collaboratively with trading partners, the client gains a greater ownership of events, reduces lead times, improves communication, and reduces costs through the supply chain.

Functional and Technical Capabilities

The system offers the following functional and technical capabilities:

- The web-based collaborative architecture enables everyone in the supply chain to have secure access to the same information, improving visibility, providing one version of the truth, and allowing for the proactive management of projects.
- The system's flexible, template-based process allows the client to determine the key events, dependencies, owners, and lead times that are used to manage processes. The system thus provides for both a mechanism for business process re-engineering and a consistent process approach throughout the internal and external community.
- Sophisticated dependency handling among events provides flexibility and allows clients to determine the complexity they would like to reflect within the track management process.
- Nested track functionality provides the flexibility to track all events with varying priorities within the same tool. For example, a parent track might be used to track the high-level events related to a process, and the nested child track could be linked to the parent track and used to track the lower-level events that also require tracking and visibility.
- The system's mass change capability provides an efficient means of maintaining track details, improving the overall accuracy and timeliness of the data tracked within the solution.
- Automated and manual diary entries provide improved controls and accountability over project management data. The ability to automatically send the diary entry as an email improves timely communication.

- By centralizing key event data in one place, the system allows the client to perform the following:
  - Evaluate the performance of events, tracks, and trading partners.
  - Report event management progress accurately.
  - Make strategic, fact-based decisions.
    - Because the system utilizes some features of the Java 2 Enterprise Edition (J2EE) architecture, clients have a choice in their selection of databases and application servers.
    - Market-proven and industry-standard application programming interfaces (API) are utilized (for example, RMI, JDBC, and so on).
    - Java applications such as Oracle Retail WebTrack have enhanced portability which means the Oracle Retail clients are not ‘locked’ into a single platform. Upgrades are easier to implement, and hardware is easier to change.
    - The WebTrack Administration Console provides the following functionality and more:
  - Options
    Oracle Retail WebTrack provides Event Options. One option allows the blanking of revised dates on an event if the revised date is the same as the planned date. A second option is the enabling of an event confirmation column in the Track Details screen.
  - Mail
    The mail administration is used to configure automatic email alerts to users based on various events.
  - Lists
    Extra fields support configuration of non-standard elements in the integration of Oracle Retail Design to Oracle Retail WebTrack.
  - Users
    Oracle Retail WebTrack provides a local type in the WebTrack user permissions to enable a button to launch Oracle Retail Design from the Track List or Track Details screens.
  - WebTrack User Console provides the following functionality and more:
  - Diary entry
    A ‘Track Name Changed’ diary entry is introduced for changes made to a track name through the integration configuration between Oracle Retail Design and Oracle Retail WebTrack.
  - Track details
    When enabled by the administrator, a column is available in the Track Details screen to show the confirmation of a plan or revised date by an event owner.
  - Track List Screen
    When enabled by the administrator and mapped through the Oracle Retail Design configuration file, the user is able to add the season attribute and the design vendor (Extra field) to the Track List screen. In addition, Oracle Retail WebTrack allows the user to launch Oracle Retail Design from the Track List screen.
  - Mail
    When added to a mail message, the user is able to launch the Oracle Retail Design application from the mail message via the Oracle Retail Design URL.

A Template-Based Approach

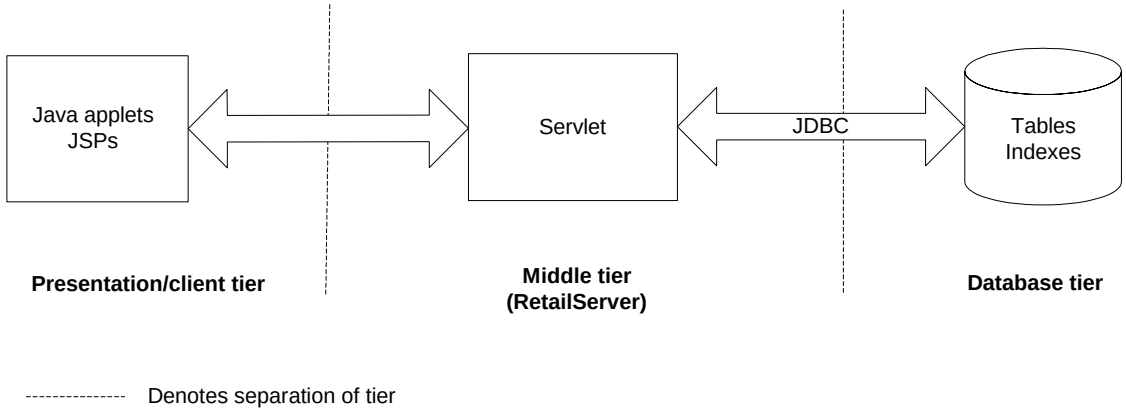
The client defines a template to include specific events and associated lead times. Events are specific tasks that the client monitors for completion. Events that are commonly used together may be grouped into a template. Using the template and supporting foundation data such as project or purchase order (PO) data, the client creates a 'track.' A track provides a mechanism for communicating expectations, schedules, and event assignments to all members of the supply chain including trading partners. In sum, to help save time within supply chain processes, Oracle Retail WebTrack uses tracks to manage the events associated with projects and (on a more complex level) POs. Because of the flexibility associated with Oracle Retail WebTrack, a client could use the application to support the pre-production tracking of a new product, the opening of a new store, or the production and logistics tracking of items on a purchase order.

Individual event owners are accountable for managing specific events within the track. Access to all tracks in one place improves visibility to changes. Automatic email alerts provide reminders to event owners if events become late or overdue. As updates and changes are made to the track, a diary of all activity is logged. Finally, the system's robust reporting abilities on track data allow the client to report on progress and to manage by exception.

Technical Architecture Overview

The three-tiered architecture of Oracle Retail WebTrack allows for the separation of presentation, business logic, and data, making the software cleaner and easier to maintain.

The diagram and brief explanation below offer a high-level conceptual view of the tiers. For a detailed description of this diagram, see the chapter, "Technical Architecture."



Oracle Retail WebTrack Technical Architecture

To facilitate the functional demands and complexity of a web-based application, the presentation tier facilitates robust client-side processing. Oracle Retail WebTrack provides a 'fat' client with Java applets, which enable a portion of the application's logic to be built into the client. JSPs serve as a place from which the application's Java applets can launch. No logic resides within the system's JSPs, except for that which facilitates utility processing (such as logging in, logging out, authentication, and so on). A browser is all that is need to access the application, and when a browser views a page that contains an applet, the applet's code is transferred to the client and executed by the browser's Java Virtual Machine (JVM).

The middle tier's primary responsibility is to service the requests of the client (for example, to fetch data, to store data, and so on). The client, after all, has no direct connection to the database or its tables and must rely upon the middle tier to accommodate its requests. The middle tier (in which the server resides) is comprised primarily of a single Java servlet called retailserver. The Java applets in the client tier communicate with retailserver, which, in turn, communicates with the database on behalf of the client. The retailserver is comprised of a number of different modules. Once the modules are loaded by the servlet, each is responsible for performing specific functional tasks. Specific system functionality sometimes requires the use of only a subset of these modules. Sub-modules, called addons, lie within the modules to help facilitate the system's integration functionality.

The database tier is the application's storage platform, containing the physical data (user and system) used throughout the application. The database's tables house data which is retrieved by the middle tier and then passed to the client. Oracle Retail WebTrack supports two database products, Oracle and DB2.

Potential Integration Points

The high-level diagram below shows the overall direction of the data among systems and products across the enterprise. For a detailed description of this diagram, see the chapter, "Integration Interface Dataflows."

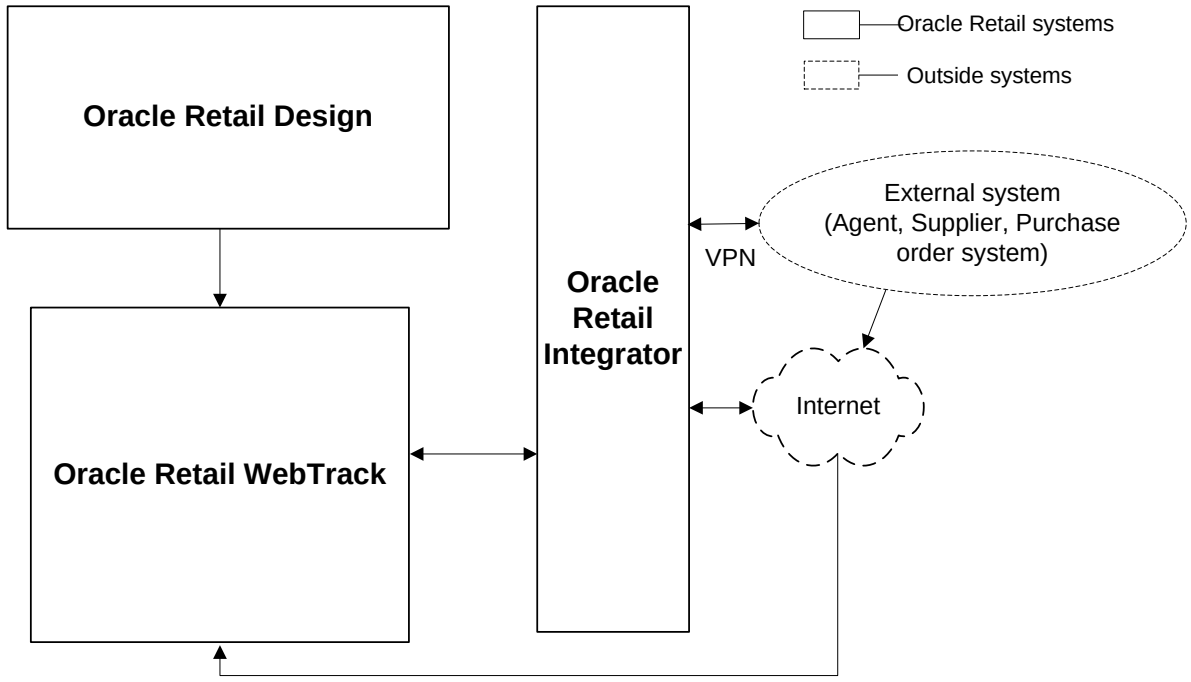
Oracle Retail WebTrack does *not* have to be integrated in order to provide a client with functionality because all of its data can be also populated through the front-end. The diagram illustrates the potential ways an external system can access the Oracle Retail systems. In production, multiple trading partners could exist, and their interfaces to the system would depend upon their individual circumstances.

When a product enters a certain stage, a project within Oracle Retail WebTrack can automatically be created using approved program details data (such as color/short description data, department name data, and so on) collected in Oracle Retail Design.

An external system can send foundation data (such as item data, division data, and so on) to Oracle Retail WebTrack to facilitate implementation and ongoing maintenance. To support daily business processes, an external system can send purchase order, event, and project data to Oracle Retail WebTrack.

Oracle Retail WebTrack can send an external system event due date data as well as track details data that includes track name, track description, events associated with the track diary entries associated with the track, and so on.

Note: Oracle Retail WebTrack, Oracle Retail Design, and Oracle Retail Integrator share the same schema.



Oracle Retail WebTrack-related dataflow across the enterprise

2
Backend System Administration

This chapter of the operations guide is intended for database administrators who provide database support and monitor the running system.

The content in this chapter is not procedural but is meant to provide descriptive overviews of key system parameters. The troubleshooting section identifies situations that might arise and their remedies.

To find further system administration information including those values which can be defined through the graphical user interface (GUI), see the Oracle Retail Server Administration Guide.

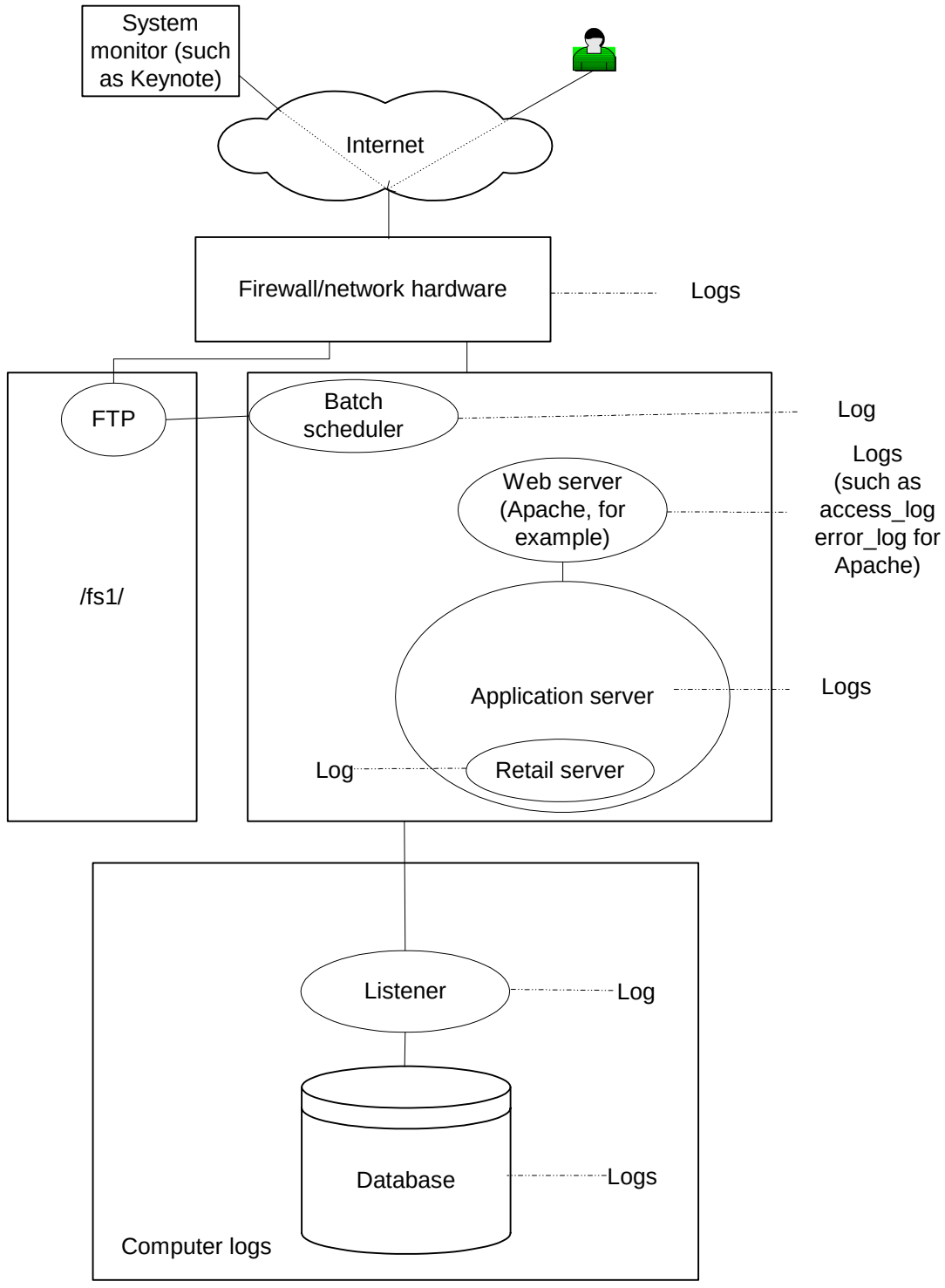
Logging and Exception Handling

Log files to be monitored exist across a number of locations throughout the system. Oracle Retail recommends that the client purchase a system monitoring tool (such as Patrol, Unicenter, and so on). Such a tool can usually be set up to send emails or pages when certain events occur in the system.

The diagram below illustrates the location of the system's log files (listed below from the bottom of the diagram up):

- Computer logs run by a system administrator
- Database logs (for Oracle or DB2)
- Listener-related log
- Retail server code log for the code running on the application server
- Application server logs (for WebSphere, BEA WebLogic, Oracle Application Server, and so on)
  For example, when WebSphere is being used, these are located in /opt/WebSphere/AppServer/logs
- Apache logs
  For example, when Apache is bundled with WebSphere, these logs are located in the following directory:
  - /opt/IBMHttpServer/logs
- Batch processing-related logs
  See the chapter, "Running Oracle Retail Integrator Batch Processes" for specific information and paths related to batch logging.
- Firewall and network hardware-related logs
- System as a whole logs (not shown on diagram)

Location of Logs Diagram



Logging locations throughout the system to be monitored

Exception Handling and General Troubleshooting

> **Note:** For specific troubleshooting suggestions, see the section, 'Troubleshooting,' later in this chapter.

When an exception arises that must be investigated, the log files listed above and shown on the diagram should be searched for errors. Errors can indicate a variety of problems, such as the following:

- The database is full or the database is having other problems.
- A filesystem is full.
- The operating system log file indicates a hardware, network file system (NFS), or other operating system error.
- The database listener is down.
- The system is having domain name system (DNS) or name resolution problems.
- There is a missing 'complete' file to initiate a batchjob.
- There is invalid xml.
- A worm overloads the firewall.
- The firewall timeouts.

A Word about a System Monitor

Oracle Retail recommends the use of a third party service (such as Keynote) that inspects the site from many different locations (sixty, for example) around the world. Such third party system monitors usually have machines scattered in different places around the world, and these machines access websites and measure their response. When the third party system monitor encounters an issue, it contacts the administrator through a page or other method.

Starting and Stopping the System Order of Operation

The following order of operation should be followed when starting the system (after, for example, application maintenance, and so on):

1. Start the applicable database network listener.
2. Start the database (such as DB2, Oracle, and so on).
3. Start the application server (such as WebSphere, and so on). Note that Apache is always started in conjunction with the application server because the two are connected.

Stopping the System

Note: A graceful shutdown of the system is always preferred. However, in the event that this method does not succeed, an administrator can force an aborted shutdown.

The following order of operation should be followed when stopping the system (for application maintenance, for example):

1. Shut down the application server. Note that Apache is always stopped in conjunction with the application server because the two are connected.
2. Shut down the database.
3. Shut down the applicable database network listener.

RetailServer Properties Documentation

A system administrator defines configurations for Oracle Retail WebTrack in a number of properties files. Most key configurable system parameters contained in these files are described in this chapter. For example, when clients build the code to their environment, they must update many of these values to their specific settings. Parameters within the system's properties files that are *not* changed (except perhaps under exceptional circumstances) are *not* described in this chapter.

Note that within a property file (and thus in some of the examples from that file below), a # sign that precedes a value in the properties file signifies that what follows is a comment and is not being utilized as a setting.

db.properties

This file contains configuration values related to the system's database.

common.prop.db
This value defines the database the system is utilizing. This value is set to one of the following:

- oracle
- db2

Oracle Parameters

- common.prop.oracle.sid
  This value defines the database name the system is utilizing. SID stands for system identifier.
- common.prop.oracle.host
  This value refers to the database listener. This value defines the 'host:port' that the database listener is utilizing.
- common.prop.user
  This value defines the '`username/password`' of the database.

DB2 Parameters

- common.prop.db2.db
  This value defines the database name the system is utilizing.
- common.prop.db2.host
  This value defines the 'host:port' that the database is utilizing.
- common.prop.user
  This value defines the '`username/password`' of the database.

mailserver.properties

This file contains configuration values related to the system's use of email. The system uses these settings in the following circumstances:

- when sending emails related to batch process status (successful, failed, and so on).
- when sending notification emails related to WebTrack timeline tracking functionality (for example, an email could be sent informing users that Thursday at noon, a step must occur).
- common.prop.mailfrom
  This value defines the sender of email sent by Oracle Retail Design or Oracle Retail WebTrack. When a user views the email, this value appears in the 'From' field.
- mailhost
  This value defines the server's name or IP address that Oracle Retail Design or Oracle Retail WebTrack uses as a SMPT mail server.
- mailtest
  This value is used to redirect mail to a deadend account rather than to the intended recipient. This setting is designed as a precaution, so that users do not get confused by bogus email from a test system. The following table explains potential values and their results:

Value	Result
#mailtest=deadend@mydomain.com	Because the value is commented out, all mail works.
mailtest=deadend@mydomain.com	All mail is redirected to the deadend account. Nobody receives mail.
mailtest=deadend@mydomain.com,+@mydomain.com	All mail is sent to the deadend address *and* any mail going to within mydomain.com is sent to the correct addresses. For testing purposes, copies of all the mail being sent are thus available for inspection at the deadend account.

Value	Result
mailtest=deadend@mydomain.com,+john.doe@oracle.com	All mail is sent to the deadend address *and* any mail going to john.doe@oracle.com is sent to the correct address.

mailtestfile

By establishing the parameter, the client designates a custom alternate properties file that contains the mailtest value. This setting is used so that changes can be made 'on the fly'. If this parameter is used, changes to values do *not* require that the server be bounced. Note that a mailtest value in the yourfilename.properties file overrides any value established in the mail.properties file. If applicable, this value is set to the following:

- mailtestfile=yourfilename.properties

main.properties

The system's module definitions are provided in this file. Each module definition consists of a class name and any addon submodules that the module may need. Some modules may not be loaded, depending upon licensing. For an understanding of the role that modules and addons play within the application, see the chapter, "Technical Architecture."

Other important settings within this file include the following:

- base
  Because this value must correspond to the context root setup of where the application was installed, this setting should *not* be changed in this file once Oracle Retail WebTrack is implemented. For example, this value could be one of the following:
  - '`/`' for a production system
  - '`/test1`' for the first of several test systems on a single physical computer
- jvmlogfile
  Path and logfile name in which to log information about the jvm and browser used for each connection to the system.
- jvmlogformat
  Sets additional formatting for the entries in the jvmlogfile. The format was 'inspired' by the logging formats in Apache. "%a" is the client IP address and "%t" is time (date/time). The remaining values are standard Java system properties on the client.
  - The default is %a [%t] "%{java.vendor}" "%{java.version}" "%{os.name}" "%{os.version}" "%{os.arch}"
  - A common entry is: jvmlogformat=+ "%{enterprisename}" "%{userfullname}"

- securemode
  This value determines whether the system forces connections to switch from http (non-secure) to https (secure) upon logon. For example, this value can be one of the following:
  - '1' forces connections to switch from http (non-secure) to https (secure) upon logon.
  - '0' does *not* force connections to switch from http (non-secure) to https (secure) upon logon.

security.properties

trackeradmin.prop.adminhosts
This file value contains the list of IP addresses that are allowed to log in to the exchange administration console. The list is a comma-separated list of address or address/mask values. If no values are entered, all hosts can connect to the administration console (*not recommended*).

warp.properties

Every night, the application sends email. The values in this file ensure that only one web server sends out email. Note that changes made to this setting *require* a bounce.

> **Note:** Only one of the web servers should be instructed to process the mail run.

- monitor
  A value of '1' instructs the web server to process the midnight mail run. A value of '0' instructs the web server *not* to process the mail run.

rcapps.properties

Values in this file change the GUI. This file thus allows you an easy way to override certain aspects of the GUI. For example, if you would like a testing system to have a different look than the production system, these settings could be used.

Colors at www.yourdomain.com are defined by the values below. Some values are in hexadecimal (Hex), which represent red, green, blue (RGB) values but in a form that browsers can interpret. Hex is a notation which mixes letters and numbers and allows three digit numbers to be expressed as two digit numbers. For example, a green background would be (0, 255, 0) in RGB, and 00FF00 in Hex. The # sign preceeding the value signifies that the value is in Hex.

- appbanner.bg
  Main banner backgroup color (default is #E13128)
- appmenu.bg
  Menu banner background (default is #F9F8C7)
- apphdr.bg
- Header background (default is #E13128)
- securemode.allow
  Set to "*" to enable securemode on all clients, or "*, !Mac" to enable securemode on all clients except those using a Macintosh. Secure mode must be disabled on Mac OS X systems if you are using a test SSL certificate because the 1.3.1 Java VM does not function without a valid host certificate.

- text.fg
  Main test color (default is #E13128)
- webmeter.allow
  The webmeter in the GUI provides an approximate visual indication of the quality of the network connection to the server. Set to "`*`" to display webmeter for all clients, or "`*, !Mac`" to display webmeter for all clients except those using a Macintosh. The webmeter is disabled for Macintosh systems because it can cause issues with JDK 1.3.1 on Mac OS X system.

Integrator.properties

maxfilesize
This value instructs the system to only accept import XML files below a given number in megabytes (for example, 128 megabytes).

license.bin

This file defines the applications that a client has licensed. Oracle Retail supplies this file when the client receives the system. Should a client choose to add a new application to the license, this file would have to be updated. Once updated, the new application would become available upon a system restart.

Troubleshooting

Issue

I am uploading a file through Oracle Retail Integrator, but system says the file is too big.

Response
Adjust the 'maxfilesize' parameter in integrator.properties. See the 'Integrator.properties' passage earlier in this chapter.

Issue

I am receiving a Remote Method Invocation (RMI) error in a batch process.

Response
Determine whether any properties refer to 'localhost'. If so, 'localhost' must be resolvable by the system. For UNIX, add a record '`127.0.0.1 localhost`' to /etc/hosts. Also, the web server (such as Apache) must listen on 'localhost'. This action is accomplished by adding the directives '`Listen 80`' and '`Listen 443`' to the conf file (for example, 'httpd.conf'). The conf file is the main parameter file for Apache. See the Oracle Retail Server Installation Guide for more information about conf file settings for the application.

Issue

When I browse to http://www.mydomain.com, the login page is visible. However, when I browse to https://www.mydomain.com, the following error appears: 'This page cannot be displayed'. In addition, one or both of the following occurs:

- An error in the Apache log file says 'handshake failed, invalid date'.
- When logging in, a certificate warning appears.

Response:
Your secure socket layer (SSL) protocol certificate may have expired. Renew it and install the new certificate from VeriSign. For more information about SSL, see your application server documentation and/or Apache documentation.

Issue

When I browse to http://www.mydomain.com, the login page is visible. However, when I browse to https://www.mydomain.com, the following error appears: 'This page cannot be displayed'. However, neither of the following occurs:

- An error in the Apache log file says 'handshake failed, invalid date'.
- When logging in, a certificate warning appears.

Response:
The web server does not have the uniform resource locator (URL) and port added as a virtual host. For example, you may need to configure the virtual host to listen for '`*:443`' or '`www.mydomain.com:443`' or '`a.b.c.d:443`' depending upon which of these URLs you would like to use to access the system.

Issue

When I browse to http://www.mydomain.com, strange characters appear on the screen.

Response
An incorrect configuration in the Apache conf file (httpd.conf) may be causing Apache to use the SSL (https) protocol on port 80, which should *not* use SSL. Ensure that the SSLEnable directive only appears inside the '<VirtualHost :443>' directive.

Issue

> **Note:** This issue is addressed with the assumption that WebSphere is the application server that you are using.

When I perform a 'ps -ef |grep -i WebpShere', two processes are shown running rather than the usual one. Is something wrong?

Response
There is probably nothing wrong. You or someone else may have the admin console open. By closing the admin processes and then performing a 'ps -ef |grep -i WebpShere', the normal number of processes appears.

Issue

When I browse to http://www.mydomain.com, I see 'Internal Server Error 500' or 'my custom error document'. A similar error also appears on the Apache 'CustomLog' file.

Response
For reasons that cannot always be ascertained, the Apache front-end can sometimes lose its connectivity to the rest of the application server. The quickest way to restore the system is to bounce the application server. If this issue occurs frequently, examine all system log files and try to identify any errors that may coincide with the instability. Reevaluate the patch level of all software including the operating system. Also, consider a cold boot of all hardware to re-initialize the entire system. See the section, 'Starting and Stopping the System,' earlier in this chapter.

Issue

Errors occur related to the reading of data that is stored on an NFS filesystem shared by more than one application server.

Response

For reasons that cannot always be ascertained, NFS filesystems can sometimes become corrupt. This corruption is often accompanied by an NFS error in the operating system log files. In addition, sometimes the same directory, when viewed from different servers, displays inconsistent data. The quickest way to restore the system may be to unmount and remount the NFS filesystems. If this issue occurs frequently, examine all system log files and try to identify any errors that may coincide with the instability. Re-evaluate the patch level of all software including the operating system. Also, consider a cold boot of all hardware to re-initialize the entire system. See the section, 'Starting and Stopping the System,' earlier in this chapter.

Issue

The following exception is displayed soon after I log in: 'SQL exception…'.

Response

The wrong database and/or the incorrect machine name could be specified in the db.properties file. To resolve the issue, correct the entries in the db.properties file, and stop and restart the application server (such as WebSphere). See the sections, 'Starting and Stopping the System' and 'db.properties' earlier in this chapter.

Issue

The following message is displayed soon after I log in: 'Virtual Host not defined … unable to connect using port 443'.

Response

Note: This issue is addressed with the assumption that WebSphere is the application server that you are using.

In the WebSphere admin GUI, select Environment->Virtual Hosts->default Host->New:

1. Enter Port = 443 and Host= *
2. Apply and save the changes you have made.
3. Stop and restart WebSphere.

Issue

First, I log into mydomain.com and click on the WebTrack. I then select the track option from the next window. My issue is that the tracks window takes a very long time to be displayed.

Response

If the user does *not* need to create tracks, the user can be defined as 'maintenance only', so that the tracks button loads more quickly. Thus, to reduce the access times, follow these steps:

1. Log in as the admin for the user.
2. Click **Webtrak** and select **user**.
3. Click the **users** tab and edit the applicable user.
4. Click the **permissions** tab and select **WebTrack**.
5. Under available types, select 'track maintenance only' and click the right arrow. 'Track maintenance only' is now one of the selected types.
6. Log in as the user modified above and access the tracks. Access times should be significantly reduced.

3
Technical Architecture

This chapter describes the overall software architecture for Oracle Retail WebTrack. The chapter provides a high-level discussion of the general structure of the system, including the various layers of Java code. This information is valuable in the following scenarios, among others:

- Troubleshooting the system.
- Implementing the system for a different application server or database.

For those who are less familiar with Java terminology, a description of Oracle Retail WebTrack-related Java terms and standards is provided for your reference at the end of this chapter.

Overview

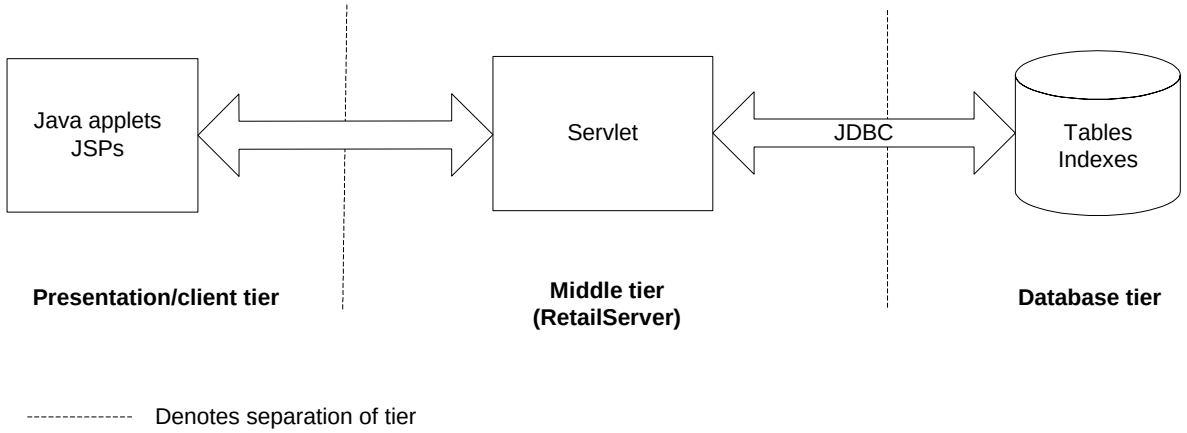
The Oracle Retail WebTrack architecture is built upon a tiered model. That is, layers of the application communicate with one another through an established hierarchy and are only able to communicate with neighboring tiers.

The application is divided into a presentation tier, a middle tier, and a database tier. The application's Java architecture has the following advantages, among others:

- The separation of presentation, business logic, and data makes the software cleaner, more maintainable, and easier to modify.
- The look and feel of the application can be updated more easily because the GUI is not tightly coupled to the backend.
- A tiered architecture has become an industry standard.
- Market-proven and industry-standard application programming interfaces (API) are utilized (for example, RMI, JDBC, and so on).
- Java applications have enhanced portability which means you are not 'locked' into a single platform. Upgrades are easier to implement, and hardware is easier to change.

The Tiered Model

The diagram below, together with the explanations that follow, offer a high-level conceptual view of the tiers and their responsibilities within the architecture. Key areas of the diagram are described in more detail in the sections that follow.



The Oracle Retail WebTrack three-tier architecture

Presentation Tier

This tier handles the presentation of the application, including its user interface. The GUI is responsible for presenting data to the user and for receiving data directly from the user through the 'front end.'

To facilitate the functional demands and complexity of a web-based application, the front end facilitates robust client-side processing. Oracle Retail WebTrack provides this 'fat' client with Java applets, which enable a portion of the application's logic to be built into the client. When a browser views a page that contains an applet, the applet's code is transferred to the client and executed by the browser's JVM.

The presentation tier is comprised in part of lightweight JSPs (JSP 1.1 specification). These JSPs serve as a place from which the application's Java applets can launch. No logic resides within the system's JSPs, except for that which facilitates utility processing (such as logging in, logging out, authentication, and so on). For a definition of a JSP, see the section, 'Oracle Retail WebTrack-Related Java Terms and Standards,' in this chapter.

Some of the system's applets are responsible for launching parts of the applications. The table below illustrates these applets and is provided for your reference. Note that they should *not* require modification.

Applet	Page	Description
uk.co.webtrak.galaxy.EnterpriseCoreAdmin	applications/core/admin.jsp	Core administration
uk.co.webtrak.design.DesignAdministration	applications/design/admin.jsp	Design administration
uk.co.webtrak.design.DesignUser	applications/design/user.jsp	Design user
uk.co.webtrak.specs.SpecAdministration	applications/specs/admin.jsp	Spectrum
uk.co.webtrak.warp.EnterprisePathAdmin	applications/tracks/admin.jsp	WebTrack administration
uk.co.webtrak.warp.WarpUserApplet	applications/tracks/user.jsp	WebTrack user
uk.co.webtrak.galaxy.SystemAdministration	phantasm/admin.jsp	System admin

Resource Bundles

The technical infrastructure of Oracle Retail WebTrack supports some languages other than English. The system includes resource bundles, which are Java files related to the internationalization process. They contain translatable text.

Middle Tier

The middle tier's primary responsibility is to service the requests of the client (for example, to fetch data, to store data, and so on). The client, after all, has no connection to the database or its tables and must rely upon the middle tier to accommodate its requests. For example, a user could initiate a request for a list of seasons through the presentation tier (in which the GUI resides). The server is responsible for getting the list by generating a query for the applicable table in the database tier. The middle tier talks with the database via the industry-standard JDBC protocol. For a definition of JDBC, see the "Oracle Retail WebTrack-Related Java Terms and Standards" section in this chapter. The server is responsible for returning these results to the client.

The middle tier, in which the server resides, is comprised primarily of a single Java servlet called retailserver. A servlet is a Java program that runs in conjunction with a web server (such as WebSphere). The Java applets in the client tier communicate with retailserver, which, in turn, communicates with the database on behalf of the client.

Retailserver contains code that is not related directly to any particular application (such as Oracle Retail Design, Oracle Retail WebTrack, and so on). This functionality includes the following:

- Handling security functionality by providing basic authorization and authentication functionality during user logon.
- Handling processing associated with logging in and logging out.

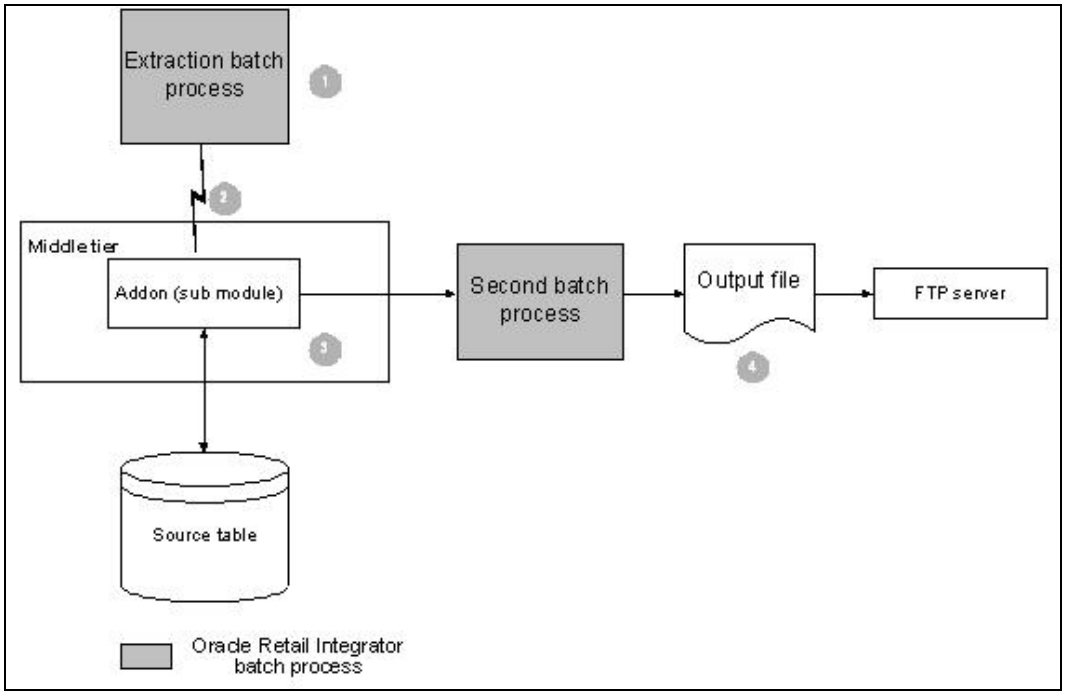
Modules and Addons

The retailserver is comprised of a number of different modules. The modules which are loaded are controlled by the main configuration file, main.properties. The main servlet class, CgiWsp, loads modules as defined in the main configuration file.

Once the modules are loaded by the servlet, each is responsible for performing specific functional tasks. Specific system functionality sometimes requires the use of only a subset of these modules.

Sub-modules lie within the modules to help facilitate the system's integration functionality. In the properties file, these sub-modules are called addons.

The modules within retailserver handle requests from the client, while the addons handle requests from within the installed application to extract and/or load data for integration purposes. The diagram below shows the role of the addons in a typical data extraction integration process. Explanations of each numbered item on the diagram follow it.



The addon's role in integration processing

1. The batch scheduler 'kicks off' an Oracle Retail Integrator batch process on the UNIX server in the production system. The Oracle Retail Integrator batch process knows the location of the yourdomain.com server, and the batch process contains parameters that instruct it about the following:
   a. what function it is performing
   b. which enterprise it is addressing
2. Using a remote method call via Java RMI, the batch process calls the addon within the middle tier of the application (such as Oracle Retail WebTrack).
3. The addon performs its task, retrieving the data and sending it to a second Oracle Retail Integrator process.
4. The second Oracle Retail Integrator process outputs the data into a file and ensures that this file is sent to the FTP site. From the FTP site, the data can be collected by the third party.

Abstract Objects to Streamline the Code

Some objects in the middle tier abstract portions of the actual persistence mechanism that is being used to persist business objects. These objects are almost all associated with the integration piece of the application, and they are never sent back to the client. Rather, once they accumulate the data from the database, they present methods to get the data which the rest of the server code can utilize. They are entirely internal to the server's processing and are described here for informational purposes *only*.

Database Specific Code in the Middle Tier

Some classes in the middle tier address the differences between the functional aspects of databases. These wrapper classes are used to access data in a database-specific way. The code within these wrapper classes 'knows' about a specific database in a generic way and is used for more complex operations such as inner joins, outer joins, sequences, and so on. Note that this code does *not* know or address specific tables in particular.

Handling Java Objects Stored in Tables

To streamline the code, in some cases the middle tier performs technical (as opposed to functional) transformations in the data it channels between the database and the client. Within the database, most tables store data in a straightforward manner (as strings, numbers, and so on). There are a few tables, however, that store Java objects within their columns. These objects, which are often shared by the client, appear as binary data within the database.

The steps below illustrate an example of a technical transformation in the middle tier. Once a specification (spec) sheet is loaded in XML in Oracle Retail Design, the middle tier performs the following steps:

1. Parses the XML.
2. Converts the XML to a Java object that represents the spec sheet.
3. Stores the Java object in the database.

When the client asks for the specification (spec) sheet, the middle tier retrieves the Java object and presents it for the client, which interprets it for the user.

Database Tier

The database tier is the application's storage platform, containing the physical data (user and system) used throughout the application. The database's tables house data which is retrieved by the middle tier and then passed to the client. This tier is not involved in the manipulation or in the delivery of the data. This tier responds to queries; it does not initiate them.

Oracle Retail WebTrack supports two database products, Oracle and DB2.

Oracle Retail WebTrack-Related Java Terms and Standards

Oracle Retail WebTrack is deployed using some of the features associated with J2EE technologies, methods, versions and/or design patterns. Key terms are defined in this section.

Applet

Java is used to write applets, which are programs designed to be executed from within another application. Applets are copied from the application server to the client machine and run within a browser. Web browsers equipped with JVM interpret applets. Applets are designed to be both cross-platform compatible and highly secure.

Instantiate

In the context of Oracle Retail WebTrack, to 'instantiate' means to create a new instance of a class.

Java 2 Enterprise Edition (J2EE)

The Java standard infrastructure for developing and deploying multi-tier applications. Implementations of J2EE provide enterprise-level infrastructure tools that enable such important features as database access, client-server connectivity, and security.

Java Database Connection (JDBC)

JDBC is a means for Java-architected applications such as Oracle Retail WebTrack to execute SQL statements against an SQL-compliant database, such as Oracle. Part of Sun's J2EE specification, most database vendors implement this specification. JDBC provides the support that allows Oracle Retail WebTrack to submit SQL queries to the database and receive the result set for further processing.

Java Development Kit (JDK), version 1.3.1

Standard Java development tools from Sun Microsystems.

Java Server Pages (JSP)

JSPs are part of Sun's J2EE specification. JSP pages appear in the Web browser as files with a .jsp extension. In the context of Oracle Retail WebTrack, JSPs merely provide a launching place for applets.

Java Virtual Machine (JVM)

JVM is a specification for software that that runs compiled Java programs. Because all Java programs are compiled for the JVM, the JVM must be implemented on a particular platform before compiled Java programs can run. The JVM allows Java to be portable because it is designed to run on most platforms.

Naming conventions in Java

- Packages: The prefix of a unique package name is always written in all-lowercase letters.
- Classes: These descriptive names are unabbreviated nouns that have both lower and upper case letters.
- Interfaces: These descriptive names are unabbreviated nouns that have both lower and upper case letters. The first letter of each internal word is capitalized.
- Methods: Methods begin with a lowercased verb. The first letter of each internal word is capitalized.

Persistence

Data is persisted when it is stored on a permanent medium. Persisted data survives the application's termination and the shut down of the computer.

Remote Method Invocation (RMI)

RMI offers a simple and direct model for distributed computation with Java objects. RMI is the action of invoking a method of a remote interface on a remote object. A method invocation on a remote object has the same syntax as a method invocation on a local object.

Servlet

A servlet is a Java platform technology that allows a web application easier access to server side resources. A servlet is a precompiled Java program that is executed on the client side. Servlets run in conjunction with a server. A servlet is usually executed in response to a request from a client browser.

Skeleton

The skeleton understands how to communicate with the stub across the RMI link. The skeleton performs all of the following:

- 'Talks' with the stub.
- Reads the parameters for the method call from the link.
- Makes the call to the remote service implementation object, accepts the return value, and then writes the return value back to the stub.

Stub

The stub contains information that allows it to connect to a remote object, which contains the implementation of the methods. The stub implements the same set of remote interfaces as the remote object's class. Once the client making the RMI call has a stub object from the RMI server, the stub talks to the skeleton code in the server using RMI protocols.

4

Integration Interface Dataflows

This chapter provides an overview of how Oracle Retail WebTrack can integrate on a functional level with other systems.

Overview

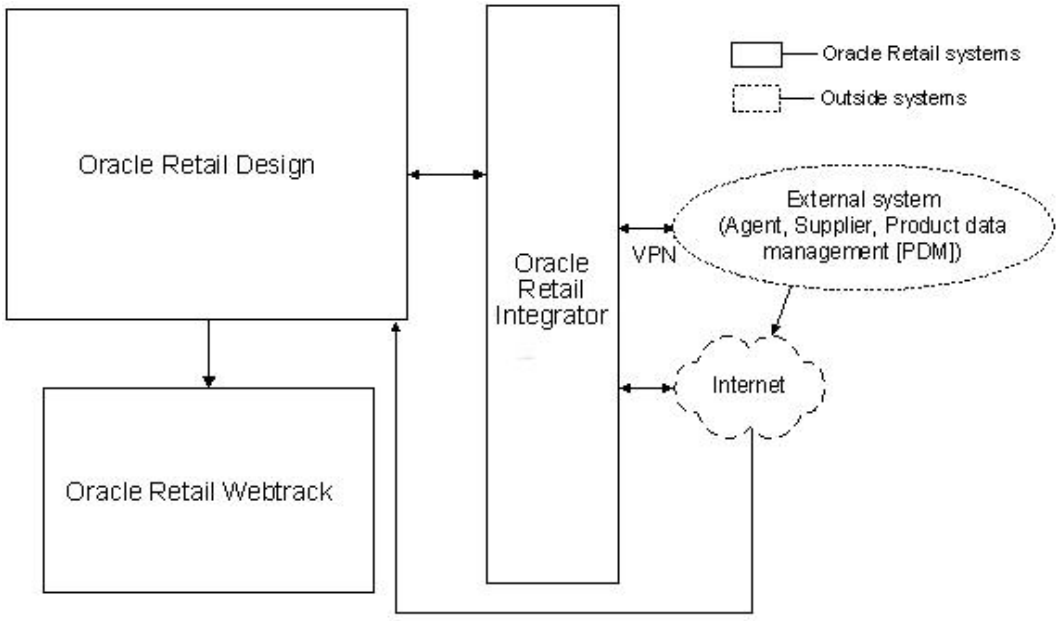
This chapter provides you with a diagram illustrating the systems that Oracle Retail WebTrack interfaces with as well as the overall dataflow among the products. The chapter is intended for those clients who utilize the Oracle Retail WebTrack integration capabilities. Oracle Retail WebTrack does *not* have to be integrated in order to provide a client with functionality because all of its data can also be populated through the front-end.

The accompanying explanations are written from a system-to-system perspective, illustrating the movement of data. Note that this discussion focuses on the functional use of data; the means of data movement (for example, batch processing) and other sites associated with those means (for example, the FTP server) are not illustrated in this chapter.

Note: Oracle Retail WebTrack, Oracle Retail Design, and Oracle Retail Integrator share the same schema.

System-to-System Dataflow

Note: The diagram illustrates the potential ways an external system can access the Oracle Retail systems. In production, multiple trading partners could exist, and their interfaces to the system would depend upon their individual circumstances.



Oracle Retail WebTrack dataflow across the enterprise

From Oracle Retail Design to Oracle Retail WebTrack

Approved product details for automatic project creation data
When a product enters a certain stage, a project within Oracle Retail WebTrack can automatically be created using approved program details data collected in Oracle Retail Design. Once Oracle Retail WebTrack creates a project, the client can create a track from the project that includes the data originating in Oracle Retail Design. Approved product details can include the following:

- Color/short description data
  Oracle Retail Design supports products (a shirt, for example). The color data represents the color of the product, and the short description data depicts the product (for example, silk, short-sleeved camp shirt). Oracle Retail WebTrack uses this data as its project name. This data is mandatory. The project integration between the two systems supports tracking by color or style.
- Department name data
  Oracle Retail WebTrack uses this data as the department name. This data is mandatory.
- Order required by date data
  This date is calculated within Oracle Retail Design based upon the initial availability date – transit time (freight on board [FOB] lead days). This data is mandatory.
- Design identifier data
  Oracle Retail WebTrack uses this data as its project number. Note that because the same Oracle Retail Design ID may exist, the project number may not be unique.
- Sell price multiplied by the quantity (per color) data
  Oracle Retail WebTrack uses this data element as the value.
- Comments (created from the product file by the Oracle Retail Design user) data
  These creation comments identify the Oracle Retail Design user who triggered the creation of the project. Oracle Retail WebTrack uses this data as comments.

- Unique style ID data
  Oracle Retail WebTrack uses this data as part of its attributes. Note that these fields are not visible to the user within Oracle Retail WebTrack until a track is created. This data creates a unique relationship between data and allows the Oracle Retail WebTrack project to be linked to the specific product/color file in Oracle Retail Design. Once a project is created using data from Oracle Retail Design, the project key identifying the new Oracle Retail WebTrack project is referenced in the Oracle Retail Design database.
- Supplier data
  Oracle Retail WebTrack uses this data as parts of its attributes. Note that these fields are not visible to the user within Oracle Retail WebTrack until a track is created.

From an External System to Oracle Retail WebTrack

Note: The DTD and/or XSD files associated with this dataflow are illustrated in the Appendix of this document, "DTD and XSD files used in Oracle Retail Integrator processing".

Foundation Data

In ordinary circumstances, this data is initially populated during implementation and maintained as necessary from season to season.

- Color palette data
  Color palette data is sent to Oracle Retail WebTrack within the palettecolor XML file. This data movement allows color palettes to be created, updated, or deleted within Oracle Retail WebTrack. Once colors are created within Oracle Retail WebTrack, they can be assigned to items and specific purchase orders within Oracle Retail WebTrack.
- Item data
  Item details data, including item description, product number, vendor, colors, value, hierarchy information, and comments are sent to Oracle Retail WebTrack within the item XML file. This data movement allows items to be created, updated, or deleted within Oracle Retail WebTrack. Once items are created within Oracle Retail WebTrack, they can be manually assigned to purchase orders.
- Division data
  Division data is sent to Oracle Retail WebTrack within the division XML file. This data movement allows divisions to be created, updated, or deleted within Oracle Retail WebTrack.

Process-Related Data

This data can flow among systems during daily business processes.

- Purchase order data
  PO details data, including order number, item number, color, quantity, vendor, and hierarchy information, are sent to Oracle Retail WebTrack within the order XML file. This data movement allows PO details to be created, updated, or deleted within Oracle Retail WebTrack.
- Event data
  Event data is sent to Oracle Retail WebTrack within the event import XML file. This data movement allows track details to be automatically updated by an external system. Specific event updates can be included within the event XML file. Specifically, dates can be completed or revised, and diary entries can be made by a defined user for a specific event within a track. Additional details are included within the XML file for validation, error notification, and logging.
- Project data
  Project details data, including the project name and number, department, value, completion date, and comments are sent to Oracle Retail WebTrack within the project XML file. This data movement allows projects to be created, updated, or deleted within Oracle Retail WebTrack. Once projects are created, they can be used to create tracks.

From Oracle Retail WebTrack to an External System

Note: The DTD and/or XSD files associated with this dataflow are illustrated in the Appendix of this document, "DTD and XSD files used in Oracle Retail Integrator processing".

- Event due date data
  When event updates occur to track due dates, Oracle Retail WebTrack sends the changed event data to the trading partner via the event due dates export XML file. This data movement helps in verifying completion dates and ensures consistency across systems.
- Track details data
  Oracle Retail WebTrack sends track details data (track name, track description, events associated with the track, diary entries associated with the track, and so on) that has changed since a specific date or since the last time the extract process was run.

5
How to Build a Test System

This chapter provides a high-level illustration of how to create a test system using a production system as a source. For example, a production system called yourdomain.com could be reproduced as a test system called test.yourdomain.com. A client might wish to build a test system for a number of reasons, including the purchase of a new computer, the testing of new code, training, and so on.

Refresh Process

The following steps illustrate the refresh process: how to copy a production system and build a test system. A system includes filesystem data, the schema within the database, and the J2EE application code (held within an .ear file).

> **Note:** To ensure consistent data between the production system and the test system, copy the filesystem data near the time that the database is copied.

1. Set up the hardware for the test system.
   Procure separate hardware, such as the disk, database, and application server, for the test system that is similar to or identical to the production system.
2. Create an empty database that is similar to the one in production.
   The test system should have equivalent tablespace structure, but many of the values in the parameter file for the database (memory, sort space, and so on) should be scaled down because the load will most likely not be equivalent to production.
3. Install the application server in the test system (including the Apache front-end piece) in the same way that it was installed for production. See the Oracle Retail Server Installation Guide for more information. Note that after a fresh installation of the application server, adjustments to values may have to be made in the following places (though for the most part they should retain the values of the production system):
   a. Ports
   b. Mime-types
   c. The Apache plug-in
   d. The Apache conf file
   e. The start script (for example, startServer.sh for WebSphere)
   f. The JAVA libraries
4. Install the production J2EE ‘war’ file on the test system application server. This production J2EE ‘war’ file is available on the Oracle Retail-packaged CD-ROM or on the production system.
5. Browse to test.mydomain.com and utilize the Oracle Retail Private Exchange Setup utility. Type in applicable data (properties files settings, and so on) in the utility about the test system. Bounce the server code (by stopping and starting the server).

> **Note:** You now have an empty system. You must now stop the application server and replace the database and file system with a copy of production.

6. Copy the production database schema using a database dump file. Oracle clients can note the following example command:
   ```
   exp userid=system/manager owner=\(retailserver\) log=exp.log consistent=y
   file=retail.dmp
   ```
   See your database vendor's applicable documentation for more information.
7. Remove all objects from the databse schema. Replace them with the exported contents of production. Oracle clients can note the following example command:
   ```
   imp userid=system/manager fromuser=\(retailserver\) touser=\(retailserver\)
   log=imp.log file=retail.dmp
   ```
8. Remove the data directory (for example, /retailserver/data). Replace it with a copy of the system. One may already exist in your backup software. If not, a tar command may be used. For example:
   ```
   tar -cvf retail_data.tar /retailserver/data
   ```
9. Extract the filesystem copy into the test system with a command. For example,
   ```
   tar -xvfp retail_data.tar
   ```
10. Start up the application server.
11. Test a logon by accessing www.test.mydomain.com and entering your username and password.
12. Change the super user password so that it is different than it is for the production system if you wish enhanced additional security. To make this change, follow these steps:
    a. Browse to http://www.yourdomain.com/phantasm.
    b. Log on.
    c. Click users.
    d. Select Super Administrator.
    e. Enter the new password in both boxes.
    f. Click Save.
13. Change all user passwords so that they are different than they are for the production system if you wish enhanced additional security. To make this change, follow these steps:
    a. Connect to the database.
    b. Enter the following SQL statement:
    ```
    UPDATE ent_users SET password = 'test';
    commit;
    ```

6

Running Oracle Retail Integrator Batch Processes

This chapter provides the following:

- An overview of the batch architecture
- A description of how to run batch processes
- A list of the command line parameters used in batch processing
- A functional summary of each batch process, along with its dependencies
- A description of some of the features of the batch processes (batch logging, return values in the GUI, and so on)
- A list of other documentation related to configuring and running Oracle Retail Integrator batch processes

Batch Process Architectural Overview

Oracle Retail Integrator is the mechanism for sharing data between Oracle Retail WebTrack and other systems through multiple channels (primarily FTP) using XML.

Note the following characteristics of the Oracle Retail Integrator batch processes:

- They are run in Java.
- They are scheduled by the client.
- Although key parameters are established through the Oracle Retail WebTrack GUI, batch processes must be run through the command line, not the GUI.
- They are designed to process large volumes of data.
- Oracle Retail recommends that they be executed during ‘off-hours’ (that is, during a time when users are not in the system such as nights).

Import and Export File-Based Batch Processes

Although Oracle Retail Integrator supports a variety of functional processes, the system is responsible for two primary mechanisms, import and export. These import and export mechanisms process a variety of XML formats. The steps involved in each mechanism are consistently applied regardless of the functional process or application supported.

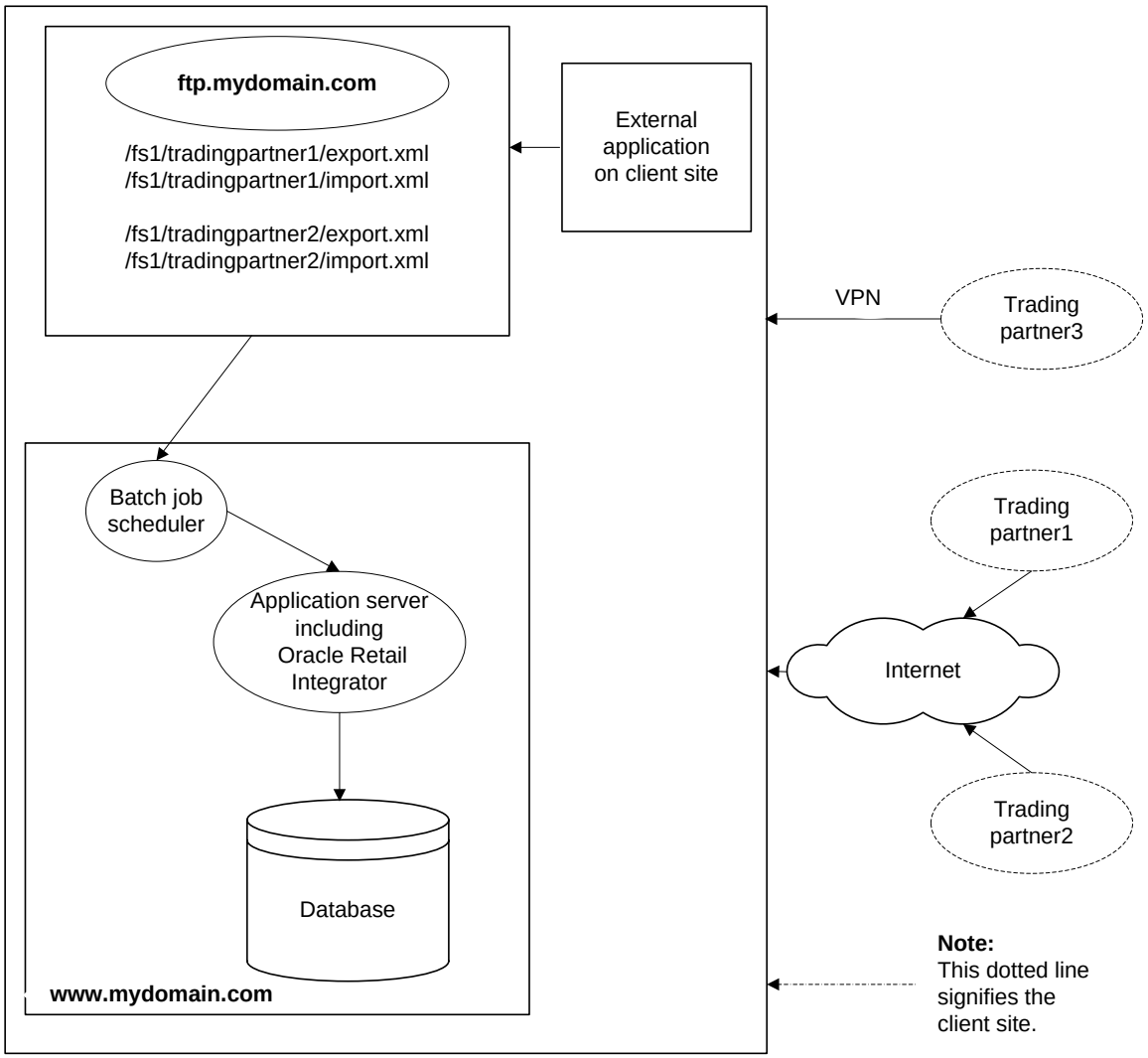
Import Processing Description and Diagram

This description and the diagram that follows it provide a brief overview to the import process.

File Transfer Protocol (FTP) is used as a communication channel for sharing files between systems and Oracle Retail Integrator. The external system (for example, an external application on the client site, a trading partner, and so on) connects in as a user to access the FTP site. The trading partner then downloads the file for the client site to the FTP directory (for example, /fs1/trading partner1/import.xml, where fs1 stands for file system 1). Note that a trading partner *not* on the client’s site uses the internet or a virtual private network (VPN) encryption-based connection.

Once Oracle Retail Integrator receives the file, the file is validated and stored as a data set before it is processed. The XML file is then translated for internal processing between Oracle Retail Integrator and the applicable application (such as Oracle Retail WebTrack).

The XML file is parsed and individual records of data are shared with the applicable application to validate and perform defined updates. The applicable application processes the data and returns the status (including success, failure, or processed with errors). Once all processing is complete, Oracle Retail Integrator sends the applicable email notifications and stores the results within Oracle Retail Integrator for viewing through the GUI or the log file.



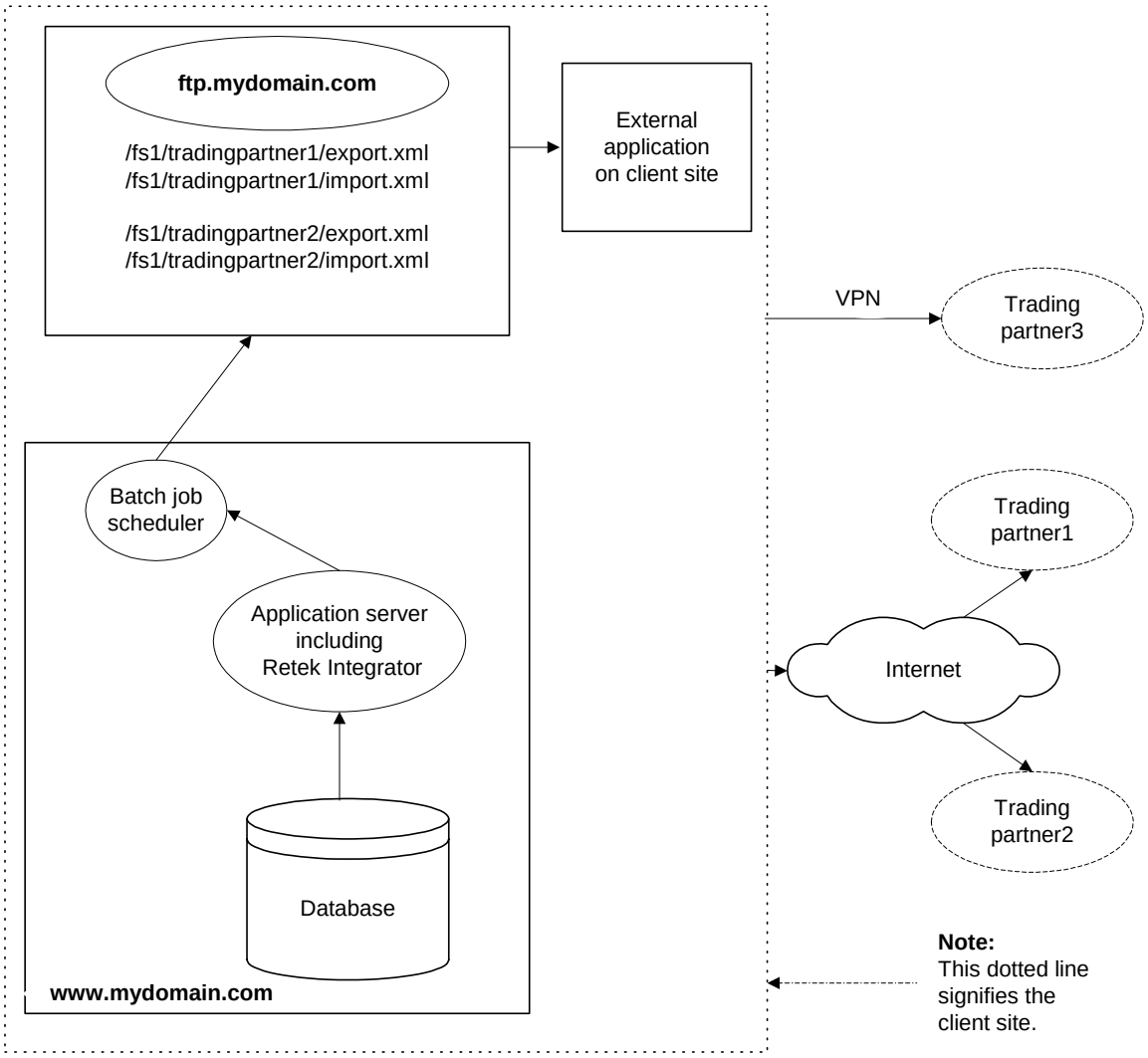
Batch import flow overview

Export Processing Description and Diagram

This description and the diagram that follows it provide a brief overview to the export process.

A scheduled batch process triggers the export process to begin processing. Based on a set of rules, application-specific processes determine the data to be included within the specific export process. When the applicable batch process runs, a file is extracted from the database. Oracle Retail Integrator is called for processing. Oracle Retail Integrator generates the supporting XML files, stores the entire data set contents in the form of an XML file, and generates the applicable success or failure responses.

A second ‘companion’ batch job runs placing the file into an FTP directory such as /fs1/trading partner1/export.xml, where fs1 stands for file system 1. The trading partner is the company with whom the client is collaborating. The trading partner has a computer that uses the internet or a VPN encryption-based connection to access the FTP site and connect in as a trading partner user. The trading partner can then perform whatever tasks it chooses (printing, uploading, and so on).



Batch export flow overview

Running a Batch Process

Executable Korn Shell Script

Java processes are scheduled through a hard-coded executable Korn shell script. This .ksh file is called cronexe.ksh. Oracle Retail provides this Korn shell script during installation (located in the 'configuration' directory you specified in the setup page). The script performs the following internally:

- Sets up the Java runtime environment variables before the Java process is run.
- Triggers the Java batch process in conjunction with the command line parameters that follow it.

Scheduler and the Command Line

If the client uses a scheduler, arguments are placed into the scheduler.

If the client does not use a scheduler, arguments must be passed in at the Unix command line.

The installation creates cronexe.ksh *without* the execute permission. To execute cronexe.ksh, you must either grant the Unix execute permission (for example, `chmod u+x cronexe.ksh`), or preface the call to cronexe.ksh with 'ksh' (for example, `ksh cronexe.ksh…`).

For example,

```
/yourpathtoconfigdir/cronexe.ksh java com.retail.integrator.BatchStep0 -i 9999 -c
BatchReport -r XMLTrackDetailExtract
1>/yourpathtodatadir/ftp/9999/export/XMLTrackDetailExtractBatchStep0.out
```

Summary of Command Line Parameters

The following is a list of command line parameters for each of the Oracle Retail WebTrack batch processes. The commands 'run' a batch job.

In the command line parameters shown below, the top line specifies the class (for example, BatchStep0) and the arguments determining which batch process runs. The line beneath that beginning with 1> specifies the directory location of the output file that accompanies the batch process. When a script is run and the batch process does *not* proceed to the point where Oracle Retail Integrator is involved, the system directs errors (which are probably significant) to the designated output file. The output file is a logical place to begin searching for an issue when you are troubleshooting batch process run problems. The location of the output file can be configured according to the client's needs.

The –i value (9999 in the example below) represents the client's enterprise database unique ID. Preexisting Oracle Retail clients cannot change this value.

Batch Process	Type	Command Line Parameters
Organization Import	Import	/yourpathtoconfigdir/cronexe.ksh java com.retail.integrator.FTPStep0 -i 9999 -d import -r DivisionImport 1>/yourpathtodatadir/ftp/9999/import/DivisionsFTPStep0.out 2>&1
Item Import	Import	/yourpathtoconfigdir/cronexe.ksh java com.retail.integrator.FTPStep0 -i 9999 -d import -r ItemImport 1>/yourpathtodatadir/ftp/9999/import/ItemsFTPStep0.out 2>&1
Projects Import	Import	/yourpathtoconfigdir/cronexe.ksh java com.retail.integrator.FTPStep0 -i 9999 -d import -r ProjectImport 1>/yourpathtodatadir/ftp/9999/import/ProjectFTPStep0.out 2>&1
Color Palette Import	Import	/yourpathtoconfigdir/cronexe.ksh java com.retail.integrator.FTPStep0 -i 9999 -d import -r PaletteColorImport 1>/yourpathtodatadir/ftp/9999/import/ColorsFTPStep0.out
Event Import	Import	/yourpathtoconfigdir/cronexe.ksh java com.retail.integrator.FTPStep0 -i 9999 -d import -r EventsImport 1>/yourpathtodatadir/ftp/9999/import/crontab.out
Purchase Order Import	Import	/yourpathtoconfigdir/cronexe.ksh java com.retail.integrator.FTPStep0 -i 9999 -d import -r WebTrackAutoImOrder 1>/yourpathtodatadir/ftp/9999/import/OrderImport.out
Track Details Export	Export	/yourpathtoconfigdir/cronexe.ksh java com.retail.integrator.BatchStep0 -i 9999 -c BatchReport -r XMLTrackDetailExtract 1>/yourpathtodatadir/ftp/9999/export/XMLTrackDetailExtractBatchStep0.out
Track Details FTP	Export	/yourpathtoconfigdir/cronexe.ksh java com.retail.integrator.FTPStep0 -i 9999 -d exportNoVal -r TrackDetailExport 1>/yourpathtodatadir/ftp/9999/export/TrackDetailExport.out
Event due dates export and FTP	Export	/yourpathtoconfigdir/cronexe.ksh java com.retail.integrator.BatchStep0 -i 9999 -c BatchExport -r WebTrackDueDateExport 1>/yourpathtodatadir/ftp/9999/export/EventExport.out

Batch Process	Type	Command Line Parameters
Partner Import	General	/yourpathtoconfigdir/cronexe.ksh java com.retail.integrator.FTPStep –I 9999 –d import –r PartnerImport 1>/yourpathtodatadir/ftp/9999/import/PartnerFTPStep0.out 2>&1
Purge	General	/yourpathtoconfigdir/cronexe.ksh java com.retail.integrator.IntegratorPurgeStep0 -e all -pt all -ft all -ld 1 1>/yourpathtoconfigdir/purgeStep0_regular.out 2>&1

Functional Descriptions and Dependencies

The following tables summarize the Oracle Retail WebTrack batch processes and their dependencies. A functional description of each batch process is provided, along with the file, where applicable, that is involved in the batch processing.

Note: XSD stands for XML Schema Definition, and DTD stands for Document Type Definition. In the context of Oracle Retail Integrator, both DTDs and XSDs are schema specification methods for XML documents.

Import Batch Processes

Batch Process (and Name Used in Command Line)	Functional Description	File	Batch Dependencies (Run Before/After)
Item import (ItemImport)	This process allows items to be created, updated, or deleted within Oracle Retail WebTrack. Item details including item description, style number, vendor, colors, value, hierarchy information, and comments are included within the item XML file. Once items are created within Oracle Retail WebTrack, they can be manually assigned to purchase orders.	item.xsd	No dependencies. Can be run independent of any other batch process.

Batch Process (and Name Used in Command Line)	Functional Description	File	Batch Dependencies (Run Before/After)
Organization import (DivisionImport)	This process allows divisions to be created, updated, or deleted within Oracle Retail WebTrack.	division.xsd	No dependencies. Can be run independent of any other batch process.
Partner import (PartnerImport)	Oracle Retail WebTrack allows for a new trading partner to be imported. Once the client is set up with a set number of user licenses (configured through Enterprise Administration), the client can submit an XML file containing the new trading partner's name, code, address and contact information. This trading partner is then automatically set up as an enterprise in yourdomain.com.	enterprisepartner.dtd	No dependencies. Can be run independent of any other batch process.
Projects import (ProjectImport)	This process allows projects to be created, updated, or deleted within Oracle Retail WebTrack. Project details including project name and number, department, value, completion date, and comments are included within the project XML file. Once projects are created, they can be used to create tracks.	project.xsd	No dependencies. Can be run independent of any other batch process.

Batch Process (and Name Used in Command Line)	Functional Description	File	Batch Dependencies (Run Before/After)
Color palette import (PaletteColorImport)	This process allows color palettes to be created, updated, or deleted within Oracle Retail WebTrack. Once colors are created within Oracle Retail WebTrack, they can be assigned to items and specific purchase orders within Oracle Retail WebTrack.	palettecolor.xsd	No dependencies. Can be run independent of any other batch process.
Event import (EventsImport)	This process allows track details to be automatically updated by an external system. Specific event updates can be included within the event import XML file. Specifically, dates can be completed or revised, and diary entries can be made by a defined user for a specific event within a track. Additional details are included within the XML file for validation and error notification and logging.	event.dtd	No dependencies. Can be run independent of any other batch process.

Batch Process (and Name Used in Command Line)	Functional Description	File	Batch Dependencies (Run Before/After)
PO import (WebTrackAutoImOrder)	This process allows PO details to be created, updated, or deleted within Oracle Retail WebTrack. PO details including order number, item number, color, quantity, vendor, and hierarchy information are included within the PO XML file. If specific foundation data included within the PO details has yet to be set up, this process also supports the creation of this foundation data. For example, if an item on the PO is not yet in the system, this batch process creates the item for the PO.	order.xsd	No dependencies. Can be run independent of any other batch process.

Export Batch Processes

Batch Process (and Name Used in Command Line)	Functional Description	File	Batch Dependencies (Run Before/After)
Track details export (XMLTrackDetailExtract)	This export process generates an XML file containing details about each track that, based on the parameter configuration, are selected to appear within this extract. The administrator has the ability to select the season or set of seasons that should be used as a parameter for this extract and a time frame for determining which product files should be selected. Specifically, the enterprise administrator can configure the process to pick up all track details that have changed since a specific date or since the last time the extract process is run. Once the XML file is generated within Oracle Retail Integrator, a separate FTP process is used to transfer data to specific directories on an FTP server.	tracks.dtd	Must be run prior to running Track details FTP.
Track details FTP (TrackDetailExport)	This batch process moves the files from the application directory to the FTP server.	tracks.dtd	Must be run after running Track details export.

Batch Process (and Name Used in Command Line)	Functional Description	File	Batch Dependencies (Run Before/After)
Event due dates export (WebTrackDueDate Export)	As event updates occur to track due dates, these changes are captured within an XML file and exported. This helps the client verify that key completion dates are consistent across the organization. The system uses the reports.dtd to select event updates for the extract (the XML files go in the warpreports directory). The file is used as validation for an input file to the event due dates export for Oracle Retail WebTrack. This file is uploaded and used during the event due dates export to identify what events should be considered within the event due dates export.	eventreport.dtd reports.dtd	No dependencies. Can be run independent of any other batch process.

General Batch Processes

Batch Process (and Name Used in Command Line)	Functional Description	File	Batch Dependencies (Run Before/After)
Purge (IntegratorPurge)	The purge script purges data in the file system based on parameters entered in Enterprise Administration. The script includes the following parameters: enterprise ID, purge type, and file types. If a file processed through Oracle Retail Integrator is purged, the file no longer is accessible through the Oracle Retail Integrator GUI. It is deleted from the file system.	N/A	No dependencies. Can be run independent of any other batch process.

Validation Processing

As part of some of its import processing, Oracle Retail Integrator performs validation. Once Oracle Retail Integrator converts the incoming XML to a class structure, Oracle Retail Integrator validates the XML to the schema, making sure that the data is formatted according to the schema. If the data is determined to be valid, it is passed along to the application (such as Oracle Retail Design or Oracle Retail WebTrack) which performs additional business rules validation (checking to determine whether this department exists in the database, for example) before submitting the data to the database. Data that Oracle Retail Integrator determines is not valid is returned to an XML format (that is still valid against the schema but tagged with an error). Note that Oracle Retail Integrator processes data piece by piece, so that only ‘bad’ data is tagged with an error (one division out of twenty in the division XML file, for example). The user can access this data, make corrections, and retry the import process.

Summary of Batch Logging

The following table associates batch processes with the type of log files they create. A description of the location of the log file is also included. Note that apart from errors located within the UNIX output file, most errors can be viewed through the applicable folder in the Oracle Retail Integrator GUI. Although folder paths are included in the notes within the table below, also see the Oracle Retail Integrator User Guide.

Batch Process (and Name Used in Command Line)	Type of Log File	Log File Location (Including Front-End References Where Applicable)
Organization import (DivisionImport)	This Unix script output file contains any errors generated through the execution of the script.	/yourpathtodatadir/ftp/9999/import/DivisionsFTPStep0.out (where 9999 is replaced by the enterprise_id) **Note:** This file *cannot* be viewed within the Oracle Retail Integrator application.
	This log file contains the general processing steps for the FTP process. This content is identical to that in the email notification sent to users identified for this run type in Exchange Administration.	/yourpathtodatadir/import/9999/FTPimportLog.txt_DATASETID.txt (where 9999 is replaced by the enterprise_id and DATASETID is a unique data set ID assigned to the file) **Note**: This file can also be viewed by clicking the FTPimportLog.txt file link seen in the /FTP/Import/Log folder in Oracle Retail Integrator.
	This log file contains the errors the XML file generates when an import is 'Processed with Errors'.	/yourpathtodatadir/import/9999/error/FILENAME.xml_DATASETID.xml (where 9999 is replaced by the enterprise_id, FILENAME is the import filename and DATASETID is a unique data set ID assigned to the file) **Note:** This file can also be viewed by clicking the Errors.xml file link seen by clicking the radio button to the left of the filename that was 'Processed with Errors' in the /WebTrack/Organization/AutoImport folder in Oracle Retail Integrator.
Item import (ItemImport)	This Unix script output file contains any errors generated through the execution of the script.	/yourpathtodatadir/ftp/9999/import/ItemsFTPStep0.out (where 9999 is replaced by the enterprise_id) **Note**: This file *cannot* be viewed within the Oracle Retail Integrator application.

Batch Process (and Name Used in Command Line)	Type of Log File	Log File Location (Including Front-End References Where Applicable)
	This log file contains the general processing steps for the FTP process. This content is identical to that in the email notification sent to users identified for this run type in Exchange Administration.	/yourpathtodatadir/import/9999/FTPimportLog.txt_DATASETID.txt (where 9999 is replaced by the enterprise_id and DATASETID is a unique data set ID assigned to the file) **Note:** This file can also be viewed by clicking the FTPimportLog.txt file link seen in the /FTP/Import/Log folder in Oracle Retail Integrator.
	This log file contains the errors the XML file generates when an import is 'Processed with Errors.'	/yourpathtodatadir/import/9999/error/FILENAME.xml_DATASETID.xml (where 9999 is replaced by the enterprise_id, FILENAME is the import filename and DATASETID is a unique data set ID assigned to the file) **Note:** This file can also be viewed by clicking the Errors.xml file link seen by clicking the radio button to the left of the filename that was 'Processed with Errors' in the /WebTrack/Items/AutoImport folder in Oracle Retail Integrator.
Projects import (ProjectImport)	This Unix script output file contains any errors generated through the execution of the script.	/yourpathtodatadir/ftp/9999/import/ProjectFTPStep0.out (where 9999 is replaced by the enterprise_id) **Note:** This file *cannot* be viewed within the Oracle Retail Integrator application.
	This log file contains the general processing steps for the FTP process. This content is identical to that in the email notification sent to users identified for this run type in Exchange Administration.	/yourpathtodatadir/import/9999/FTPimportLog.txt_DATASETID.txt (where 9999 is replaced by the enterprise_id and DATASETID is a unique data set ID assigned to the file) **Note:** This file can also be viewed by clicking the FTPimportLog.txt file link seen in the /FTP/Import/Log folder in Oracle Retail Integrator.

Batch Process (and Name Used in Command Line)	Type of Log File	Log File Location (Including Front-End References Where Applicable)
	This log file contains the errors the XML file generates when an import is 'Processed with Errors.'	/yourpathtodatadir/import/9999/error/FILENAME.xml_DATASETID.xml (where 9999 is replaced by the enterprise_id, FILENAME is the import filename and DATASETID is a unique data set ID assigned to the file) **Note:** This file can also be viewed by clicking the Errors.xml file link seen by clicking the radio button to the left of the filename that was 'Processed with Errors' in the /WebTrack/Projects/AutoImport folder in Oracle Retail Integrator.
Color palette import (PaletteColorImport)	This Unix script output file contains any errors generated through the execution of the script.	/yourpathtodatadir/ftp/9999/import/ColorsFTPStep0.out (where 9999 is replaced by the enterprise_id) **Note:** This file *cannot* be viewed within the Oracle Retail Integrator application.
	This log file contains the general processing steps for the FTP process. This content is identical to that in the email notification sent to users identified for this run type in Exchange Administration.	/yourpathtodatadir/import/9999/FTPimportLog.txt_DATASETID.txt (where 9999 is replaced by the enterprise_id and DATASETID is a unique data set ID assigned to the file) **Note:** This file can also be viewed by clicking the FTPimportLog.txt file link seen in the /FTP/Import/Log folder in Oracle Retail Integrator.

Batch Process (and Name Used in Command Line)	Type of Log File	Log File Location (Including Front-End References Where Applicable)
	This log file contains the errors the XML file generates when an import is 'Processed with Errors.'	/yourpathtodatadir/import/9999/error/FILENAME.xml_DATASETID.xml (where 9999 is replaced by the enterprise_id, FILENAME is the import filename and DATASETID is a unique data set ID assigned to the file) **Note:** This file can also be viewed by clicking the Errors.xml file link seen by clicking the radio button to the left of the filename that was 'Processed with Errors' in the /WebTrack/Palette Colors/AutoImport folder in Oracle Retail Integrator.
Event import (EventsImport)	This Unix script output file contains any errors generated through the execution of the script.	/yourpathtodatadir/ftp/9999/import/crontab.out (where 9999 is replaced by the enterprise_id) **Note:** This file cannot be viewed within the Oracle Retail Integrator application
	This log file contains the general processing steps for the FTP process. This content is identical to that in the email notification sent to users identified for this run type in Exchange Administration.	/yourpathtodatadir/import/9999/FTPimportLog.txt_DATASETID.txt (where 9999 is replaced by the enterprise_id and DATASETID is a unique data set ID assigned to the file) **Note:** This file can also be viewed by clicking the FTPimportLog.txt file link seen in the /FTP/Import/Log folder in Oracle Retail Integrator.

Batch Process (and Name Used in Command Line)	Type of Log File	Log File Location (Including Front-End References Where Applicable)
	This log file contains the errors the XML file generates when an import is 'Processed with Errors.'	/yourpathtodatadir/import/9999/error/FILENAME.xml_DATASETID.xml (where 9999 is replaced by the enterprise_id, FILENAME is the import filename and DATASETID is a unique data set ID assigned to the file) **Note:** This file can also be viewed by clicking the Errors.xml file link seen by clicking the radio button to the left of the filename that was 'Processed with Errors' in the /WebTrack/Events/AutoImport folder in Oracle Retail Integrator.
PO import (WebTrackAutoImOrder)	This Unix script output file contains any errors generated through the execution of the script.	/yourpathtodatadir/ftp/9999/import/OrderImport.out (where 9999 is replaced by the enterprise_id) **Note:** This file *cannot* be viewed within the Oracle Retail Integrator application.
	This log file contains the general processing steps for the FTP process. This content is identical to that in the email notification sent to users identified for this run type in Exchange Administration.	/yourpathtodatadir/import/9999/FTPimportLog.txt_DATASETID.txt (where 9999 is replaced by the enterprise_id and DATASETID is a unique data set ID assigned to the file) **Note:** This file can also be viewed by clicking the FTPimportLog.txt file link seen in the /FTP/Import/Log folder in Oracle Retail Integrator.

Batch Process (and Name Used in Command Line)	Type of Log File	Log File Location (Including Front-End References Where Applicable)
	This log file contains the errors the XML file generates when an import is 'Processed with Errors.'	/yourpathtodatadir/import/9999/error/FILENAME.xml_DATASETID.xml (where 9999 is replaced by the enterprise_id, FILENAME is the import filename and DATASETID is a unique data set ID assigned to the file) **Note:** This file can also be viewed by clicking the Errors.xml file link seen by clicking the radio button to the left of the filename that was 'Processed with Errors' in the /WebTrack/Orders/AutoImport folder in Oracle Retail Integrator.
Partner Import (PartnerImport)	This Unix script output file contains any errors generated through the execution of the script.	/yourpathtodatadir/ftp/9999/import/PartnerFTPStep0.out (where 9999 is replaced by the enterprise_id) **Note:** This file *cannot* be viewed within the Oracle Retail Integrator application.
	This log file contains the general processing steps for the FTP process. This content is identical to that in the email notification sent to users identified for this run type in Exchange Administration.	/yourpathtodatadir/import/9999/FTPimportLog.txt_DATASETID.txt (where 9999 is replaced by the enterprise_id and DATASETID is a unique data set ID assigned to the file) **Note:** This file can also be viewed by clicking the FTPimportLog.txt file link seen in the /FTP/Import/Log folder in Oracle Retail Integrator.

Batch Process (and Name Used in Command Line)	Type of Log File	Log File Location (Including Front-End References Where Applicable)
	This log file contains the errors the XML file generates when an import is 'Processed with Errors.'	/yourpathtodatadir/import/9999/error/FILENAME.xml_DATASETID.xml (where 9999 is replaced by the enterprise_id, FILENAME is the import filename and DATASETID is a unique data set ID assigned to the file) **Note:** This file can also be viewed by clicking the Errors.xml file link seen by clicking the radio button to the left of the filename that was 'Processed with Errors' in the /Partner Enterprise/AutoImport folder in Oracle Retail Integrator.
Track Details Export (XMLTrackDetailExtract)	This Unix script output file contains any errors generated through the execution of the script.	/yourpathtodatadir/ftp/9999/export/XMLTrackDetailExtractBatchStep0.out (where 9999 is replaced by the enterprise_id) **Note:** This file *cannot* be viewed within the Oracle Retail Integrator application.
	This log file contains the general processing steps produced when running the Track Details Export script. This content is identical to that in the email notification sent to users identified for this run type in Exchange Administration.	/yourpathtodatadir/import/9999/BatchLog.txt_DATASETID.txt (where 9999 is replaced by the enterprise_id and DATASETID is a unique data set ID assigned to the file) **Note:** This file can also be viewed by clicking the BatchLog.txt file link seen when clicking the radio button to the left of the filename in the /Batch Jobs/Reports/WebTrack/Track Details folder in Oracle Retail Integrator.

Batch Process (and Name Used in Command Line)	Type of Log File	Log File Location (Including Front-End References Where Applicable)
	This log file contains the general processing steps for creating the Distributed Track Details Export file. This content is identical to that in the email notification sent to users identified for this run type in Exchange Administration.	/yourpathtodatadir/import/9999/ReportDistLog.txt_DATASETID.txt (where 9999 is replaced by the enterprise_id and DATASETID is a unique data set ID assigned to the file) **Note:** This file can also be viewed by clicking the ReportDistLog.txt file seen when clicking the radio button to the left of the filename in the /Batch Jobs/Reports/WebTrack/Distributed Track Details folder in Oracle Retail Integrator.
Track Details FTP (TrackDetailsExport)	This Unix script output file contains any errors generated through the execution of the script.	/yourpathtodatadir/ftp/9999/export/TrackDetailExport.out (where 9999 is replaced by the enterprise_id) **Note:** This file *cannot* be viewed within the Oracle Retail Integrator application.
	This log file contains general FTP processing steps.	/yourpathtodatadir/import/9999/FTPexportLog.txt_DATASETID.txt (where 9999 is replaced by the enterprise_id and DATASETID is a unique data set ID assigned to the file) **Note:** This file can also be viewed by clicking the FTPexportLog.txt file link seen in the /FTP/Export/Log folder in Oracle Retail Integrator.
Event due dates export FTP (WebTrackDueDateExport)	This Unix script output file contains any errors generated through the execution of the script.	/yourpathtodatadir/ftp/9999/export/EventExport.out (where 9999 is replaced by the enterprise_id) **Note:** This file *cannot* be viewed within the Oracle Retail Integrator application.

Batch Process (and Name Used in Command Line)	Type of Log File	Log File Location (Including Front-End References Where Applicable)
	This log file contains general FTP processing steps.	/yourpathtodatadir/import/9999/FTPexportLog.txt_DATASETID.txt (where 9999 is replaced by the enterprise_id and DATASETID is a unique data set ID assigned to the file) **Note:** This file can also be viewed by clicking the FTPexportLog.txt file link seen in the /FTP/Export/Log folder in Oracle Retail Integrator.
	This log file contains general processing steps of the Event Due Date Export FTP script, including filenames that were exported.	/yourpathtodatadir/import/9999/AutoExLog.txt_DATASETID.txt (where 9999 is replaced by the enterprise_id and DATASETID is a unique data set ID assigned to the file) **Note:** This file can also be viewed by clicking the AutoExLog.txt file link seen in the /WebTrack/Events/Due Dates/AutoExportLog folder or AutoExLog.txt file link seen by clicking the radio button to the left of the filename in the /Batch Jobs/Reports/WebTrack/EventDue Dates folder in Oracle Retail Integrator.
	This log file contains general batch processing steps of the Event Due Date Export FTP script.	/yourpathtodatadir/import/9999/BatchLog.txt_DATASETID.txt (where 9999 is replaced by the enterprise_id and DATASETID is a unique data set ID assigned to the file) **Note:** This file *cannot* be viewed within the Oracle Retail Integrator application.
Purge (IntegratorPurge)	This Unix script output file contains any errors generated through the execution of the script.	/yourpathtodatadir/purgeStep0_regular.out **Note:** This file *cannot* be viewed within the Oracle Retail Integrator application.

Batch Process (and Name Used in Command Line)	Type of Log File	Log File Location (Including Front-End References Where Applicable)
	This log file contains general purge processing steps. This data includes what was and was not purged for the enterprise_id. This content is identical to that in the email notification sent to users identified for this run type in Exchange Administration.	/yourpathtodatadir/import/9999/purgeLog.txt_DATASETID (where 9999 is replaced by the enterprise_id and DATASETID is a unique data set ID assigned to the file) **Note:** This file can also be viewed by clicking the purgeLog.txt file link seen in the /Batch Jobs/Purge/Log folder in Oracle Retail Integrator.

Key GUI Field Descriptions Including Batch Return Values

Because backend and front end batch-related data overlap in Oracle Retail Integrator, the following key front-end field descriptions are described here as a helpful reference. See the Oracle Retail Integrator User Guide for the latest information.

Field Name	Field Description
Date	Displays the time and date that the file was sent to retail.com Displayed in GMT format: MM/DD/YY HH:MM:SS
Status	Displays one of five messages. The message describes the result of the file processing. *Success.* The operation successfully completed and no errors were encountered. *Error: NOT Processed.* Something occurred to block the file import. Try the import again or contact your site administrator. *Processing File ...* File is imported, but processing is not complete. When this message is displayed, click **Refresh** to update the status. *Processed with Errors.* The import partially succeeded. An error file containing those records that were NOT imported should are created and a link to it displayed in the Errors.xml field in the Detail frame. Save the error file to your desktop and fix any errors. Try the import again. *Error: Processing Failed.* A major problem occurred while processing the file. This could be the result of invalid XML formatting, missing required seed data, or some retail.com resource that is not available. More information is available in the Detail frame's error message field. Check file formatting and/or contact your site administrator.
Records Imported	Displays the number of records imported to the WebTrack database tables from the Oracle Retail Integrator staging tables during the file import process. This number should not be zero and should be equal to the total number of records loaded. If this is not true then an error occurred during the process.
Records Loaded	Displays the number of records loaded directly from the XML file to the Oracle Retail Integrator staging tables during the file import process. This number should not be zero and should be equal to the total number of records imported. If not, an error occurred during the process.

Field Name	Field Description
Errors.xml	Contains a link to an Errors.xml file containing records that could not be imported. This is seen only if the file status in the List frame is 'Imported with errors'. When the link is clicked on, the current Detail frame is replaced with the contents of the error file. Each record or sub-record that could not be imported will contain a non-empty <error_text> …</error_text> element that describes the error that was encountered.
Error Message	Displays any error message or messages about the file import process.

A

Appendix: DTD and XSD Files Used in Oracle Retail Integrator Processing

Note: XSD stands for XML Schema Definition, and DTD stands for Document Type Definition. In the context of Oracle Retail Integrator, both DTDs and XSDs are schema specification methods for XML documents. XSDs allow for more validation.

Import Batch Processes

division.xsd Used in DivisionImport

```
<?xml version="1.0" encoding="UTF-8"?>
<!--W3C Schema generated by XML Spy v4.3 U (http://www.xmlspy.com)-->

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
       xmlns:jaxb="http://java.sun.com/xml/ns/jaxb"
       xmlns:xjc="http://java.sun.com/xml/ns/jaxb/xjc"
       jaxb:extensionBindingPrefixes="xjc"
       jaxb:version="1.0"
       elementFormDefault="qualified">


<!-- JAXB global settings -->

        <xs:annotation>
                <xs:appinfo>
                        <jaxb:globalBindings>
                                <xjc:serializable/>
                        </jaxb:globalBindings>
                </xs:appinfo>
        </xs:annotation>

    <!-- department -->

    <xs:complexType name="departmentType">
        <xs:sequence>
            <xs:element name="departmentname" type="xs:string"/>
            <xs:element name="departmentnumber" type="xs:string"/>
        </xs:sequence>
        <xs:attribute name="action" use="required">
            <xs:simpleType>
                <xs:restriction base="xs:NMTOKEN">
                    <xs:enumeration value="A"/>
                    <xs:enumeration value="U"/>
                    <xs:enumeration value="D"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
    </xs:complexType>
```

```
    <!-- division -->

    <xs:complexType name="divisionType">
        <xs:sequence>
            <xs:element name="divisionname" type="xs:string"/>
            <xs:element name="divisionnumber" type="xs:string"/>
            <xs:element name="department" type="departmentType" minOccurs="0"
maxOccurs="unbounded"/>
            <xs:element name="error_text" type="xs:string" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="action" use="required">
            <xs:simpleType>
                <xs:restriction base="xs:NMTOKEN">
                    <xs:enumeration value="A"/>
                    <xs:enumeration value="U"/>
                    <xs:enumeration value="D"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
    </xs:complexType>


<!-- Outer divisions element -->

    <xs:element name="divisions">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="division" type="divisionType" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="enterprise" type="xs:int"/>
        </xs:complexType>
    </xs:element>
</xs:schema>
enterprisepartner.dtd Used in PartnerImport
<?xml version="1.0" encoding="UTF-8"?>
<!--DTD generated by XML Spy v4.3 U (http://www.xmlspy.com)-->
<!ELEMENT account_number (#PCDATA)>
<!ELEMENT address (address1?, address2?, city?, state_province?, postal_code?,
country_name?, phone?)>
<!ELEMENT address1 (#PCDATA)>
<!ELEMENT address2 (#PCDATA)>
<!ELEMENT administrator (first_name, last_name, user_name, password, email)>
<!ATTLIST administrator
    initial_admin CDATA #REQUIRED
>
<!ELEMENT administrators (administrator)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT company_name (#PCDATA)>
<!ELEMENT company_type (#PCDATA)>
<!ELEMENT country_name (#PCDATA)>
<!ELEMENT default_contact (#PCDATA)>
<!ELEMENT department_number (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT enterprise (company_name, enterprise_code, industry?, company_type?,
address*, services+, administrators+, users*, partnership+, errors*)>
<!ATTLIST enterprise
    user CDATA #REQUIRED
    send_mail CDATA #REQUIRED
    send_password CDATA #REQUIRED
    action CDATA #REQUIRED
>
```

```
<!ELEMENT enterprise_code (#PCDATA)>
<!ELEMENT enterprises (enterprise)>
<!ATTLIST enterprises
    version CDATA #IMPLIED
    enterprise CDATA #REQUIRED
>
<!ELEMENT first_name (#PCDATA)>
<!ELEMENT industry (#PCDATA)>
<!ELEMENT last_name (#PCDATA)>
<!ELEMENT partner_code (#PCDATA)>
<!ELEMENT partner_info (#PCDATA)>
<!ELEMENT partner_roles (service_code, permission*)>
<!ELEMENT partnership (account_number?, partner_code, default_contact?,
partner_info?, scope+, permissions*)>
<!ELEMENT password (#PCDATA)>
<!ELEMENT permission (#PCDATA)>
<!ATTLIST permission
    all_users CDATA #REQUIRED
>
<!ELEMENT permissions (partner_roles+)>
<!ELEMENT phone (#PCDATA)>
<!ELEMENT postal_code (#PCDATA)>
<!ELEMENT scope (department_number+)>
<!ELEMENT service (service_code, service_limit?)>
<!ELEMENT service_code (#PCDATA)>
<!ELEMENT service_limit (#PCDATA)>
<!ELEMENT services (service+)>
<!ELEMENT state_province (#PCDATA)>
<!ELEMENT type (#PCDATA)>
<!ELEMENT user (first_name, last_name, user_name, password, email, type?,
services*)>
<!ATTLIST user
    account_manager CDATA #REQUIRED
>
<!ELEMENT user_name (#PCDATA)>
<!ELEMENT errorMessage (#PCDATA)>
<!ELEMENT tag_name (#PCDATA)>
<!ELEMENT users (user)>
<!ELEMENT errors (tag_name, errorMessage)>
```

event.dtd Used in EventsImport

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT child_name (#PCDATA)>
<!ELEMENT color_code (#PCDATA)>
<!ELEMENT date (#PCDATA)>
<!ELEMENT events (event+)>
<!ATTLIST events
    version CDATA #IMPLIED
    enterprise CDATA #IMPLIED
>
<!ELEMENT item_number (#PCDATA)>
<!ELEMENT order_number (#PCDATA)>
<!ELEMENT parent_name (#PCDATA)>
<!ELEMENT partner_code (#PCDATA)>
<!ELEMENT project_key (#PCDATA)>
<!ELEMENT project_name (#PCDATA)>
<!ELEMENT project_number (#PCDATA)>
<!ELEMENT event (partner_code?, order?, project?, parent_name?, child_name?,
dates?, diary_comments?)>
```

```
<!ATTLIST event
    action NMTOKEN #REQUIRED
    track_type NMTOKEN #REQUIRED
    track_owner NMTOKEN #REQUIRED
>
<!ELEMENT order (order_number, item_number, color_code?, track_quantity?)>
<!ATTLIST order
    track_by NMTOKEN #REQUIRED
>
<!ELEMENT project (project_name?, project_number, project_key?)>
<!ELEMENT dates (date)>
<!ATTLIST dates
    date_type NMTOKEN #REQUIRED
    user CDATA #IMPLIED
    check NMTOKEN #IMPLIED
>
<!ELEMENT diary_comments (#PCDATA)>
<!ATTLIST diary_comments
    user CDATA #IMPLIED
>
<!ELEMENT track_quantity (#PCDATA)>
<!ATTLIST track_quantity
    user CDATA #IMPLIED
>
```

item.xsd Used in ItemImport

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
       xmlns:jaxb="http://java.sun.com/xml/ns/jaxb"
       xmlns:xjc="http://java.sun.com/xml/ns/jaxb/xjc"
       jaxb:extensionBindingPrefixes="xjc"
       jaxb:version="1.0"
       elementFormDefault="qualified">

    <!-- JAXB global settings -->

         <xs:annotation>
                 <xs:appinfo>
                         <jaxb:globalBindings>
                                 <xjc:serializable/>
                         </jaxb:globalBindings>
                 </xs:appinfo>
         </xs:annotation>


<xs:complexType name="attribute_listType">
       <xs:sequence>
           <xs:element name="attribute" type="xs:string"/>
       </xs:sequence>
   </xs:complexType>
   <xs:complexType name="color_listType">
       <xs:sequence>
           <xs:element name="color_code" type="xs:string"/>
       </xs:sequence>
   </xs:complexType>
   <xs:complexType name="itemType">
       <xs:sequence>
           <xs:element name="item_name" type="xs:string"/>
           <xs:element name="department_num" type="xs:string"/>
           <xs:element name="style_num" type="xs:string"/>
           <xs:element name="vpn" type="xs:string" minOccurs="0"/>
```

```
            <xs:element name="class_name" type="xs:string" minOccurs="0"/>
            <xs:element name="subclass_name" type="xs:string" minOccurs="0"/>
            <xs:element name="color_list" type="color_listType" minOccurs="0"
maxOccurs="unbounded"/>
            <xs:element name="unit_value" type="xs:double" minOccurs="0"/>
            <xs:element name="attribute_list" type="attribute_listType"
minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="item_comments" type="xs:string" minOccurs="0"/>
            <xs:element name="supplier" type="supplierType" minOccurs="0"
maxOccurs="unbounded"/>
            <xs:element name="error_text" type="xs:string" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="action" use="required">
            <xs:simpleType>
                <xs:restriction base="xs:NMTOKEN">
                    <xs:enumeration value="A"/>
                    <xs:enumeration value="U"/>
                    <xs:enumeration value="D"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
    </xs:complexType>
    <xs:element name="items">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="item" type="itemType" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="enterprise" type="xs:string"/>
        </xs:complexType>
    </xs:element>
    <xs:complexType name="supplierType">
        <xs:sequence>
            <xs:element name="supplier_num" type="xs:string"/>
            <xs:element name="account_manager" type="xs:string" minOccurs="0"/>
            <xs:element name="factory" type="xs:string" minOccurs="0"/>
            <xs:element name="country" type="xs:string" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="action">
            <xs:simpleType>
                <xs:restriction base="xs:NMTOKEN">
                    <xs:enumeration value="A"/>
                    <xs:enumeration value="U"/>
                    <xs:enumeration value="D"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
    </xs:complexType>
</xs:schema>
```

order.xsd Used in WebTrackAutoImOrder

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- Schema for order imports, converted from ordersschema.xdr.

     Modified to match real order XML imports as found on the
     production site.
-->

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
       xmlns:jaxb="http://java.sun.com/xml/ns/jaxb"
       xmlns:xjc="http://java.sun.com/xml/ns/jaxb/xjc"
       jaxb:extensionBindingPrefixes="xjc"
       jaxb:version="1.0"
       elementFormDefault="qualified">

    <!-- JAXB global settings -->

         <xs:annotation>
                 <xs:appinfo>
                         <jaxb:globalBindings>
                                 <xjc:serializable/>
                         </jaxb:globalBindings>
                 </xs:appinfo>
         </xs:annotation>
    <xs:complexType name="colorType">
        <xs:sequence>
            <xs:element name="color_code" type="xs:string"/>
            <xs:element name="color_desc" type="xs:string"/>
            <xs:element name="red" type="xs:int" minOccurs="0"/>
            <xs:element name="green" type="xs:int" minOccurs="0"/>
            <xs:element name="blue" type="xs:int" minOccurs="0"/>
            <xs:element name="option_qty" type="xs:double" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="action" use="required">
            <xs:simpleType>
                <xs:restriction base="xs:NMTOKEN">
                    <xs:enumeration value="A"/>
                    <xs:enumeration value="U"/>
                    <xs:enumeration value="D"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
    </xs:complexType>
    <xs:complexType name="orderType">
        <xs:sequence>
            <xs:element name="order_num" type="xs:string"/>
            <xs:element name="department_num" type="xs:string"/>
            <xs:element name="supplier_num" type="xs:string"/>
            <xs:element name="supplier_name" type="xs:string" minOccurs="0"/>
            <xs:element name="due_date" type="xs:string"/>
            <xs:element name="create_date" type="xs:string"/>
            <xs:element name="orderitem" type="orderitemType"
maxOccurs="unbounded"/>
            <xs:element name="uda" type="udaType" minOccurs="0"
maxOccurs="unbounded"/>
            <xs:element name="order_comments" type="xs:string" minOccurs="0"/>
            <xs:element name="error_text" type="xs:string" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="action" use="required">
            <xs:simpleType>
```

```
                <xs:restriction base="xs:NMTOKEN">
                    <xs:enumeration value="A"/>
                    <xs:enumeration value="U"/>
                    <xs:enumeration value="D"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="track_by" use="required">
            <xs:simpleType>
                <xs:restriction base="xs:NMTOKEN">
                    <xs:enumeration value="S"/>
                    <xs:enumeration value="O"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="order_status" type="xs:string"/>
    </xs:complexType>
    <xs:complexType name="orderitemType">
        <xs:sequence>
            <xs:element name="product_number" type="xs:string"/>
            <xs:element name="product_desc" type="xs:string"/>
            <xs:element name="vendor_part_number" type="xs:string" minOccurs="0"/>
            <xs:element name="class" type="xs:string" minOccurs="0">
                       <xs:annotation>
                                <xs:appinfo>
                                       <jaxb:property name="classId"/>
                               </xs:appinfo>
                       </xs:annotation>
                </xs:element>
            <xs:element name="subclass" type="xs:string" minOccurs="0"/>
            <xs:element name="style_qty" type="xs:double"/>
            <xs:element name="color" type="colorType" minOccurs="0"
maxOccurs="unbounded"/>
            <xs:element name="product_value" type="xs:double" minOccurs="0"/>
            <xs:element name="product_comments" type="xs:string" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="action" use="required">
            <xs:simpleType>
                <xs:restriction base="xs:NMTOKEN">
                    <xs:enumeration value="A"/>
                    <xs:enumeration value="U"/>
                    <xs:enumeration value="D"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
    </xs:complexType>
    <xs:element name="orders">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="order" type="orderType" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="enterprise" type="xs:string"/>
            <xs:attribute name="version">
                <xs:simpleType>
                    <xs:restriction base="xs:NMTOKEN">
                        <xs:enumeration value="1.1"/>
                        <xs:enumeration value="1.2"/>
                        <xs:enumeration value="2.0"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
        </xs:complexType>
```

```
    </xs:element>
    <xs:complexType name="udaType">
        <xs:sequence>
            <xs:element name="uda_code" type="xs:string"/>
            <xs:element name="uda_value" type="xs:string"/>
        </xs:sequence>
        <xs:attribute name="action" use="required">
            <xs:simpleType>
                <xs:restriction base="xs:NMTOKEN">
                    <xs:enumeration value="A"/>
                    <xs:enumeration value="U"/>
                    <xs:enumeration value="D"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="uda_type" use="required">
            <xs:simpleType>
                <xs:restriction base="xs:NMTOKEN">
                    <xs:enumeration value="FREEFORM"/>
                    <xs:enumeration value="LIST"/>
                    <xs:enumeration value="DATE"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
    </xs:complexType>
</xs:schema>
```

palettecolor.xsd Used in PaletteColorImport

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
       xmlns:jaxb="http://java.sun.com/xml/ns/jaxb"
       xmlns:xjc="http://java.sun.com/xml/ns/jaxb/xjc"
       jaxb:extensionBindingPrefixes="xjc"
       jaxb:version="1.0"
       elementFormDefault="qualified">

    <!-- JAXB global settings -->

         <xs:annotation>
                 <xs:appinfo>
                         <jaxb:globalBindings>
                                 <xjc:serializable/>
                         </jaxb:globalBindings>
                 </xs:appinfo>
         </xs:annotation>
    <xs:complexType name="palettecolorType">
        <xs:sequence>
            <xs:element name="color_name" type="xs:string"/>
            <xs:element name="color_code" type="xs:string"/>
            <xs:element name="rgb" type="xs:double"/>
            <xs:element name="seasonlist" type="xs:string" minOccurs="0"/>
            <xs:element name="scope" type="xs:string" minOccurs="0"/>
            <xs:element name="error_text" type="xs:string" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="action" use="required">
            <xs:simpleType>
                <xs:restriction base="xs:NMTOKEN">
                    <xs:enumeration value="A"/>
                    <xs:enumeration value="U"/>
                    <xs:enumeration value="D"/>
                </xs:restriction>
```

```
            </xs:simpleType>
        </xs:attribute>
    </xs:complexType>
    <xs:element name="palettecolors">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="palettecolor" type="palettecolorType"
maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="enterprise" type="xs:string"/>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

project.xsd Used in ProjectImport

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
       xmlns:jaxb="http://java.sun.com/xml/ns/jaxb"
       xmlns:xjc="http://java.sun.com/xml/ns/jaxb/xjc"
       jaxb:extensionBindingPrefixes="xjc"
       jaxb:version="1.0"
       elementFormDefault="qualified">

    <!-- JAXB global settings -->

         <xs:annotation>
                 <xs:appinfo>
                         <jaxb:globalBindings>
                                 <xjc:serializable/>
                         </jaxb:globalBindings>
                 </xs:appinfo>
         </xs:annotation>

    <xs:complexType name="projectType">
        <xs:sequence>
            <xs:element name="projectnumber" type="xs:string"/>
            <xs:element name="project_name" type="xs:string"/>
            <xs:element name="unit_value" type="xs:double"/>
            <xs:element name="completion_date" type="xs:string"/>
            <xs:element name="department" type="xs:string"/>
            <xs:element name="comments" type="xs:string"/>
            <xs:element name="error_text" type="xs:string" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="action" use="required">
            <xs:simpleType>
                <xs:restriction base="xs:NMTOKEN">
                    <xs:enumeration value="A"/>
                    <xs:enumeration value="U"/>
                    <xs:enumeration value="D"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
    </xs:complexType>
    <xs:element name="projects">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="project" type="projectType"
maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="enterprise" type="xs:string"/>
        </xs:complexType>
```

```
    </xs:element>
</xs:schema>
```

Export Batch Processes

eventreport.dtd Used in WebTrackDueDateExport

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- DTD for Warp report output.  Version 0.01y -->

<!-- Reports is the enclosing element.
     It contains one or more individual reports.
-->

<!ELEMENT Reports (Report+)>

<!-- Report contains update information for a single named report. It contains
     zero or more EventUpdate elements.

     The name is defined in the report configuration.  This allows a number of
     different reports to be defined by an enterprise.
-->

<!ELEMENT Report (EventUpdate*)>
<!ATTLIST Report name CDATA #REQUIRED>

<!-- The EventUpdate element records a single event update.  The action
     attribute defines the update type and the when attribute is the time at
     which the update occurred.  The format is 'YYYYMMDDhhmmss'.

     The nested Track and Event elements give information on the track and
     event.
-->

<!ELEMENT EventUpdate (Partner?, Track, Event)>
<!ATTLIST EventUpdate action (complete | cancel | revise | accept | reject |
replan) #REQUIRED
             when   CDATA                                          #REQUIRED>

<!-- The Partner element provides information on the supplier.  It will not be
present
     for project tracks.
-->

<!ELEMENT Partner (PartnerName, PartnerCode)>
<!ELEMENT PartnerName (#PCDATA)>
<!ELEMENT PartnerCode (#PCDATA)>

<!-- The Track element defines the track containing an updated event.  The
     type attribute defines the track type.

     The StyleNumber and VendorNumber elements will not be present for project
     tracks.
-->

<!ELEMENT Track (TrackName, Department, OrderNumber, ItemName, StyleNumber?,
VendorNumber?)>
<!ATTLIST Track type (po | project) #REQUIRED>
```

```
<!-- Track data -->

<!ELEMENT TrackName    (#PCDATA)>
<!ELEMENT Department   (#PCDATA)>
<!ELEMENT OrderNumber  (#PCDATA)>

<!-- Item data -->

<!ELEMENT ItemName     (#PCDATA)>
<!ELEMENT StyleNumber  (#PCDATA)>
<!ELEMENT VendorNumber (#PCDATA)>

<!-- The event element defines the event udpated -->

<!ELEMENT Event (EventName, EventDate)>
<!ELEMENT EventName (#PCDATA)>

<!-- The EventDate element contains the current revised date (or actual date on
     completion) of the event. The format is 'YYYYMMDD'.
-->

<!ELEMENT EventDate (#PCDATA)>
```

tracks.dtd Used in XMLTrackDetailExtract and in TrackDetailExport

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- DTD for Warp tracks export output.  Version 0.01z -->

<!-- Tracks is the enclosing element.
     It contains zero or more individual tracks.
-->

<!ELEMENT Tracks (Track*)>

<!-- The Track element contains information for a single track.  The
     type attribute defines the track type.

     The alert attribute defines the alert state of the track at the time the
     export was generated.

     The state attribute defines the track state (active, cancelled, archived).

     The suspended attribute will be true if the track has been suspended; in
     this case the SuspendDate element will also be present.

     A project track will contain a nested Project element; a PO track will
     contain a nested Order element.

     The optional OrderAttributes element contains any attributes saved with
     the order or project.

     The events element contains the track events.

     The optional Diary element contains the track Diary.  It will be present
     if requested in the export process.
 -->

<!ELEMENT Track (TrackName, CreationDate, ChangeDate, SuspendDate?, DivisionName,
DivisionNumber, DepartmentName, DepartmentNumber,
        (Project | Order), OrderAttributes?, Events, Diary?)>
```

```
<!ATTLIST Track type     (po | project)               #REQUIRED
       alert    (green | amber | red)               "green"
       state    (active | archived | cancelled) "active"
       suspended (false | true)              "false">

<!-- Basic Track data -->

<!ELEMENT TrackName    (#PCDATA)>

<!-- CreationDate, ChangeDate and SuspendDate are timestamps in the format:

    YYYYMMDDHHMMSS

     Timestamps are expressed in UTC time.
-->

<!ELEMENT CreationDate (#PCDATA)>
<!ELEMENT ChangeDate   (#PCDATA)>
<!ELEMENT SuspendDate  (#PCDATA)>

<!-- Hierarchy data -->

<!ELEMENT DivisionName   (#PCDATA)>
<!ELEMENT DivisionNumber (#PCDATA)>

<!ELEMENT DepartmentName   (#PCDATA)>
<!ELEMENT DepartmentNumber (#PCDATA)>

<!-- Project information

     ProjectInfo is the extre information saved with the project
-->

<!ELEMENT Project (ProjectName, ProjectNumber, ProjectInfo)>

<!ELEMENT ProjectName   (#PCDATA)>
<!ELEMENT ProjectNumber (#PCDATA)>
<!ELEMENT ProjectInfo   (#PCDATA)>

<!-- Order information

     The Partner element contains information on the supplier associated with
     the order.
-->

<!ELEMENT Order (OrderNumber, ItemName, Quantity, StyleNumber, VendorNumber,
OrderInfo, Partner)>

<!ELEMENT OrderNumber  (#PCDATA)>
<!ELEMENT ItemName     (#PCDATA)>
<!ELEMENT Quantity     (#PCDATA)>
<!ELEMENT StyleNumber  (#PCDATA)>
<!ELEMENT VendorNumber (#PCDATA)>
<!ELEMENT OrderInfo    (#PCDATA)>

<!-- Partner information -->

<!ELEMENT Partner (PartnerName, PartnerCode, AccountNumber, PartnerInfo)>

<!ELEMENT PartnerName   (#PCDATA)>
<!ELEMENT PartnerCode   (#PCDATA)>
<!ELEMENT AccountNumber (#PCDATA)>
```

```
<!ELEMENT PartnerInfo   (#PCDATA)>

<!-- Attribute information.  The Attribute element contains the name and value
     of a single attribute; the 'key' attribute is the application specific
     key associated with the attribute; it will be zero if the key has no
     meaning.

     Tracks associated projects or orders which were created by the Design
     application will include special attributes allowing reference back to
     the data in Design.
-->

<!ELEMENT OrderAttributes (Attribute*)>

<!ELEMENT Attribute (AttributeName, AttributeValue)>
<!ATTLIST Attribute key CDATA #REQUIRED>

<!ELEMENT AttributeName  (#PCDATA)>
<!ELEMENT AttributeValue (#PCDATA)>

<!-- Event information -->

<!ELEMENT Events (Event+)>

<!-- Information on a single event.  The alert attribute is the alert status
     of the event at the time the export was generated.  The state attribuet
     is 'active' if the event is still incomplete, 'complete' if the event has
     been completed normally, and 'cancelled' if the event has been cancelled.

     The EventOwner element defines the event owner.

     The InitialDate element will not be present for events in old tracks
     created before this information was available in the database.

     The ActualDate element will be present if the event has been completed
     or cancelled.

     The 'revisedconfirmed' attribute reflects the state of the new 'confirmed'
     flag for the event.
-->

<!ELEMENT Event (EventName, EventOwner, InitialDate?, PlanDate, RevisedDate,
ActualDate?)>

<!ATTLIST Event alert      (green | amber | red)            "green"
       state       (active | complete | cancelled) "active"
       revisedconfirmed (true | false)                     "false">

<!ELEMENT EventName (#PCDATA)>

<!-- Event owner

     The EventPartner element will be omitted if it is 'this enterprise'.
-->

<!ELEMENT EventOwner (EventPartner?, EventContact)>
<!ELEMENT EventPartner (#PCDATA)>
<!ELEMENT EventContact (#PCDATA)>
```

```
<!-- Dates

     Each is a date in the format:

    YYYYMMDD

     This represents a day in the same timezone as used internally in WebTrack.
-->

<!ELEMENT InitialDate (#PCDATA)>
<!ELEMENT PlanDate    (#PCDATA)>
<!ELEMENT RevisedDate (#PCDATA)>
<!ELEMENT ActualDate  (#PCDATA)>
<!-- Diary information.  Entries are ordered by descending date. -->

<!ELEMENT Diary (DiaryEntry*)>

<!-- Single diary entry.

     The action attribute defines one of the standard track or event actions:

    user                 entry made by user

    created              track created
    createdsuspended         track created suspended
    archived             track archived
    cancelled            track cancelled
    reinstated           track made active again
    suspended            track suspended
    unsuspended          track unsuspended
    recalculated             track recalculated with new template
    ownerchanged             track owner changed
    quantitychanged              track quantity changed
    split                track split
    namechanged          track name changed

    eventrevised             event date revised
    eventcompleted           event completed
    eventcancelled           event cancelled
    eventautocancelled       event auto cancelled
    eventaccepted            event revised date accepted
    eventrejected            event revised date rejected
    eventreplanned           event plan date changed
    eventreplannedex         event plan date and revised date changed
    eventadded           event added to track
    eventdeleted             event deleted from track
    eventownerchanged        event owner (contact) changed

    eventrevisedconfirmed         event revised date is 'confirmed'
    eventrevisedunconfirmed       event revised date is 'unconfirmed'


     The export utility allows filtering by entry action type.
-->

<!ELEMENT DiaryEntry (DiaryDate, DiaryText?, DiaryEvent?, DiaryEventDate?,
DiaryUser, DiaryEmails?)>
```

```
<!ATTLIST DiaryEntry action (user | created | createdsuspended | archived |
cancelled | reinstated | suspended | unsuspended | recalculated |
                ownerchanged | quantitychanged | split | namechanged |
eventrevised | eventcompleted | eventcancelled | eventautocancelled |
                eventaccepted | eventrejected | eventreplanned | eventreplannedex
| eventadded | eventdeleted | eventownerchanged
                eventrevisedconfirmed | eventrevisedunconfirmed) #REQUIRED>

<!-- Date/time of entry as YYYYMMDDHHMMSS stamp -->

<!ELEMENT DiaryDate (#PCDATA)>

<!-- DiaryText is the additional text for the diary entry; for 'user' actions it
is the diary text -->

<!ELEMENT DiaryText (#PCDATA)>

<!-- If the entry is event-specific, DiaryEvent is the name of the associated
     event; if present DiaryEventDate is the event revised date or actual date
     if the event is complete.
-->

<!ELEMENT DiaryEvent     (#PCDATA)>
<!ELEMENT DiaryEventDate (#PCDATA)>

<!-- DiaryUser indicates the user performing the update.
     The enterprise name is omitted if it is the local enterprise
-->

<!ELEMENT DiaryUser (DiaryUserName, DiaryUserEmail, DiaryUserEnterprise?)>

<!ELEMENT DiaryUserName       (#PCDATA)>
<!ELEMENT DiaryUserEmail      (#PCDATA)>
<!ELEMENT DiaryUserEnterprise (#PCDATA)>

<!-- Optional section containing extra email addresses -->

<!ELEMENT DiaryEmails (DiaryEmail+)>
<!ELEMENT DiaryEmail  (#PCDATA)>
```

reports.dtd Used in WebTrackDueDateExport

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- DTD for Warp report filters.  Version 0.01x -->

<!-- Reports is the enclosing element.  It contains zero or more individual
     report definitions.  The enterprise identifier is implied by the report
     source.
 -->

<!ELEMENT Reports (Report*)>

<!-- Report defines a single named report. It contains zero or more
     EventUpdate elements.
-->

<!ELEMENT Report (EventUpdate*)>
<!ATTLIST Report name CDATA #REQUIRED>
```

```
<!-- The EventUpdate element defines a event filter.  An update will be
     included in the report if it matches any of the filters.

     The action attribute defines the update type; 'any' will match all
     updates.  The single EventName element defines the event name. Named
     events only are supported - implementing an 'any' event filter is not
     possible currently because events in nested tracks do not have any direct
     reference to the top level owning enterprise; for performance reasons we
     do not want to query for the top level enterprise on every update.

     The 'replan' action is for updating of planned events via integration
     processes; it cannot be generated by user actions.
-->

<!ELEMENT EventUpdate (EventName)>
<!ATTLIST EventUpdate action (any | complete | cancel | revise | accept | reject |
replan) #REQUIRED>

<!ELEMENT EventName (#PCDATA)>
```