Administration Guide for Oracle Siebel eStatement Manager

Version 4.7

May 31, 2007

ORACLE

Contents

3 Setting up a New Application and Jobs

1 Preface

About Customer Self-Service and eaSuite

Oracle has developed the industry's most comprehensive software and services for deploying Customer Self-Service solutions. eaSuite™ combines electronic presentment and payment (EPP), order management, knowledge management, personalization and application integration technologies to create an integrated, natural starting point for all customer service issues. eaSuite's unique architecture leverages and preserves existing infrastructure and data, and offers unparalleled scalability for the most demanding applications. With deployments across the healthcare, financial services, energy, retail, and communications industries, and the public sector, eaSuite powers some of the world's largest and most demanding customer self-service applications. eaSuite is a standards-based, feature rich, and highly scalable platform, that delivers the lowest total cost of ownership of any self-service solution available.

eaSuite consists of four product families:

- Electronic Presentment and Payment (EPP) Applications
- Advanced Interactivity Applications
- Enterprise Productivity Applications
- Development Tools

Electronic Presentment and Payment (EPP) Applications are the foundation of Oracle's Customer Self-Service solution. They provide the core integration infrastructure between organizations' backend transactional systems and end users, as well as rich e-billing, e-invoicing and e-statement functionality. Designed to meet the rigorous demands of the most technologically advanced organizations, these applications power Customer Self-Service by managing transactional data and by enabling payments and account distribution.

- **eStatement Manager** is the core infrastructure of enterprise Customer Self-Service solutions for organizations large and small with special emphasis on meeting the needs of organizations with large numbers of customers, high data volumes and extensive integration with systems and business processes across the enterprise. Organizations use eStatement with its data access layer, composition engine, and security, enrollment and logging framework to power complex Customer Self-Service applications.
- **ePayment Manager**™ is the electronic payment solution that decreases payment processing costs, accelerates receivables and improves operational efficiency. ePayment Manager is a complete payment scheduling and warehousing system with real-time and batch connections to payment gateways for Automated Clearing House (ACH) and credit card payments, and payments via various payment processing service providers.

Oracle's Development Tools are visual development environments for designing and configuring Oracle's Customer Self-Service solutions. The Configuration Tools encompass data and rules

management, workflow authoring, systems integration, and a software development kit that makes it easy to create customer and employee-facing self-service applications leveraging eaSuite.

About This Guide

This guide is intended for system administrators or other IT professionals responsible for setting up and running a live eStatement Manager application in a J2EE environment. It describes the general process and specific procedures required to:

- Set up a new eStatement Manager application and the associated jobs using the eStatement Command Center. (For information about using bulk publishing to set up your application in a production environment, see the Command Center online Help.) Produce regular online statements electronically and manage the ongoing live production process.

This guide assumes you have:

- Installed eStatement Manager in your J2EE environment.
- Designed and developed the necessary application files (using DefTool and Composer in a Windows NT or 2000 environment).

This guide does *not* describe general UNIX or Windows system administration. See the appropriate UNIX or Windows user documentation.

Related Documentation

The following online Help is available in the eStatement Command Center:

Online Help	How to Access
Comprehensive Command Center Help	Click **Help** on the Command Center menu. Help contains additional information about running your application in a live production environment.
Screen-level Command Center Help	Click the **Help** button on a screen for details about that particular screen. Click **Help Contents** there to access the complete Production Help.

This guide is part of the eStatement Manager documentation set. For more information about implementing your eStatement application, see one of the following guides:

Print Document	Description
Installation Guide for Oracle Siebel eStatement Manager	How to install eStatement and configure it in a distributed environment.
Data Definition (DefTool) Guide for Oracle Siebel eStatement Manager	How to create Data Definition Files (DDFs) for use in indexing your application and extracting data for live presentment.

Print Document	Description
Presentation Design (Composer Guide) for Oracle Siebel eStatement Manager	How to create Application Logic Files (ALFs) to present statement data for dynamic online display.
Deploying and Customizing J2EE Applications Guide for Oracle Siebel eStatement Manager	How to deploy and customize the J2EE applications provided by eStatement Manager. This guide also describes how to deploy the Sample application provided by eStatement and how to validate that it is set up correctly by running a job through your installed eStatement environment.

2 Overview of the Application Setup Process

Before Getting Started

Your eStatement Manager application was created by a project team in your organization. An eStatement application consists of various design files used in a live production environment with your eStatement Manager software to enable Web users to view statements online.

During the Mastering process, your project team evaluated your organization's online presentation needs along with your data input files. They created an application that would deliver the specific data you want customers to see presented exactly the way you want them to see it. They used the eStatement Manager GUI design tools, DefTool and Composer, to create these files.

You must use the eStatement Command Center to set up and configure your application to prepare it for implementation in a live production environment.

For information about using bulk publishing to set up your application in a production environment, see the Command Center online Help. You may find it useful to print individual help topics.

Before setting up your application in the Command Center, you must:

- Become familiar with the eStatement Manager design files created for your application and the particular account information they are intended to provide the user. Creating and configuring the correct production jobs with the appropriate configuration settings requires a thorough understanding of your application. For example, if your application contains .DDF, .ALF, and template files, you must understand what each of these version sets has been designed to present. This chapter describes how each type of design file is used during production to create a particular view of statement data online.
- Work with your project team to establish what jobs you need to define and which job configuration settings you need for your application to work as intended by your design team. Review the job configuration options described in Chapter 3.

The Application Setup Process

The process of setting up a new application in the eStatement Command Center includes three basic steps. If you have multiple applications, it is best to set up one application at a time.

To set up a new eStatement application, you must:

1. **Create a new application.** This short step requires you to define, or name, the application in the eStatement Command Center, identify the data source, and specify the number of partitions to use for the Index database table.

2 **Create and configure the associated production jobs.** To implement your application in a live environment, you must configure various production jobs. This chapter describes the types of jobs you must create to make your statement data available for online presentment.

   For each job, you must choose the configuration options that will enable your eStatement application to function as intended; see Chapter 3 for a description of all configuration options. For some jobs, you must also publish associated version sets.

3 **Publish dynamic Web view files (version sets).** This chapter describes what views are and how eStatement Manager uses dynamic Web views to extract and present statements online. Dynamic Web views are discussed further in "Dynamic Web Views: HTML, XS, CSV, XML, Chart, XSLT, and XML Query" on Page 15.

Once you have defined your eStatement application, created and configured jobs, and published the dynamic Web views, you can proceed to Chapter 4 to set up a schedule for each job and begin live production. Note that eStatement does not automatically schedule jobs to run; you must manually specify job schedules for production; see Chapter 4.

You can optionally publish all version sets for dynamic Web views and batch jobs in bulk. See Command Center Help for details about bulk publishing.

What Jobs Do I Need to Create?

Each eStatement application requires certain batch jobs run on a recurring basis to make statement data available for online viewing.

The specific number and type of production (batch) jobs you need to create and configure depends on the number and type of views the project team has developed in your application.

You must create and configure the following jobs for an application:

- **An Indexer job, if your application uses live retrieval** to index the data file in preparation for live statement viewing on the web. Applications using live retrieval must have an Indexer job. An Indexer job is also necessary to enable you to generate email output.
- **An EmailNotification job, if your application has an email notification view** to send an email message to enrolled users informing them that a statement awaits them online.
- **A Purge App job** to periodically delete old data references from the index, email, report activity, detail, annotations, and dispute tables.
- **A Purge Logs job** to periodically delete old data from the logs table in the database.

Each job type is described in more detail in this chapter.

CAUTION: HTML Output or XML Output jobs are necessary only if you plan to provide users access to static output; not live retrieval. A Detail Extractor job is necessary only if you plan to upload recurring table data from the input file to a database table for merging using additional application functionality.

Why Do I Need an Indexer Job?

Each application that uses live statement retrieval requires an Indexer job to prepare the input data for dynamic viewing.

An Indexer job:

- Enables Web users to view statements using live retrieval.
- Lets you index data fields such as customer name, amount due, and due date for display on a page listing historical statements available for viewing (sometimes called a history list). If you use the eStatement Manager sub-document indexing feature, the Indexer job automatically indexes all group fields defined in the parent group of a sub-account in the DDF to enable sub-documents to appear in a history list as well.
- Enables you to generate email notifications (which you create in a separate job).
- Enables you to extract recurring table data from the input file and load it into a database table for later use by a separate tool to load the data and additional application support to use the data.

By scheduling the Indexer job to run automatically on a regular interval coinciding with the generation of your statement data file, you enable your latest statements to be routinely available for on-demand Web presentation.

The Indexer job extracts important data about your data file, called metadata, and places it in the database. When a user clicks a link to their statement, the browser uses index data in the database, the data input file, and the dynamic view files to present the user's statement.

The Indexer job also extracts data about any fields you choose to index, such as customer name, amount due, and due date, to the database. Indexing these fields makes them available for display in a historical list of bills.

The Indexer job consists of separate tasks that run sequentially; see "Creating and Configuring an Indexer Job" on Page 25 for details.

Other Production Jobs

EmailNotification Job

You must set up and configure an EmailNotification job if your application is designed to generate an email notification message. An email message informs a user that an online statement is available, and can include a direct link to dynamically view the statement.

In a live environment, you run an EmailNotification job after the Indexer job because it consists of individual tasks that depend on the successful completion of the Indexer job.

For details on setting up an EmailNotification job, see "Creating and Configuring an EmailNotification Job" on Page 31.

Note that if an EmailNotification job fails because the mail server goes down or other network-related issues, you can run the MessageFailRecovery job to resend unsent email. For details, see "Creating and Configuring a MessageFailRecovery Job" on page 35.

Purge App Job

Purge App is a system maintenance job that removes index, email, reporting, detail, annotations, dispute, and Process Workflow Controller (PWC) job and task instance data from the database. Configure this job and run it periodically to free up space on your database server and to limit user access to historical statements.

For more information about the Purge App job and other system maintenance activities, see Command Center online Help.

For details on setting up a Purge App job, see "Creating and Configuring a Purge App Job" on page 36.

Purge Logs Job

Purge Logs is a system maintenance job that removes historical information from the log table in the system database (for all applications). Configure this job and run it periodically to free up space on your database server.

For more information about the Purge Logs job and other system maintenance activities, see Command Center Help.

For details on setting up a Purge Logs job, see "Creating and Configuring a Purge Logs Job" on page 42.

HTML Output and XML Output Jobs

HTML Output and XML Output jobs create an HTML or XML output file with extracted data for each primary key. If you plan to make statements available for live retrieval on the Web, you do not need to create an HTML Output or XML Output job.

Configure an HTML Output job only if you plan to limit access to your database and let users view a static HTML output file only. Static HTML output files may be necessary if you partner with thin or thick consolidators for statement presentment. The choice to use static output would be the result of specific performance and security issues.

Configure an XML Output job if you plan to generate a static XML output file for loading into another database.

For details on setting up an HTML Output job, see "Creating and Configuring an HTML Output Job" on Page 43.

For details on setting up an XML Output job, see "Creating and Configuring an XML Output Job" on Page 45.

Detail Extractor Job

Set up and configure a Detail Extractor job only if you intend to upload data from your input file to a database table. You can use the uploaded data in any way, merging it with online statements, performing data mining, etc. You must create any applications needed to extract and use the data, however.

A Detail Extractor job is not required for presenting basic print statements on the Web using eStatement Manager.

In a live environment, you run the Detail Extractor job after the Indexer job; it only processes a data input file that the Indexer job has successfully indexed.

For details on setting up a Detail Extractor job, see "Creating and Configuring a Detail Extractor Job" on Page 48.

What is a View?

A **view** is a set of design files that results in a particular presentation of statement data.

A view can enable a user to dynamically display formatted statements live on the Web, generate email notifying users that an online statement is available, or to present other account data in various formats.

Dynamic Web Views: HTML, XS, CSV, XML, Chart, XSLT, and XML Query

A dynamic Web view is a set of design files that dynamically present a particular view of statement data to a user online. The design files identify which data to extract and how to display the data to the user.

An eStatement application can have one or more views, customized for an organization's online presentment needs. Multiple views can present different levels of statement information such as a summary page and statement detail pages.

When an enrolled user clicks a link to view their statement online, eStatement Manager uses the view files along with the application's data input file and index data from the database (generated by the Indexer job) to dynamically present the statement on the Web.

A typical **dynamic HTML Web view** consists of a pair of DDF and ALF files, and one or more associated HTML templates:

- **DDF** – A DDF is a Data Definition File, which contains the rules for finding and extracting data from your application's input data source. This DDF file is used during live statement retrieval. Your project team creates DDF files using eStatement Manager's DefTool.
- **ALF** – An ALF is an Application Logic File, which contains the rules for presenting the data extracted from the data input source in a template on the Web, in email, etc. The ALF can also contain business logic (conditional statements that consider current statement data) to display alternate messages or advertisements for marketing or business purposes. Your project team creates ALF files using the eStatement Manager Composer tool.
- **HTML Templates** – Customized HTML templates format and present the extracted data for viewing in a browser. A view can have multiple templates associated with it. Your project team creates HTML files (using any variety of methods), which they then manipulate appropriately for online presentment using the eStatement Manager Composer tool.

In addition to HTML-formatted views, eStatement application views can dynamically present data in one of these formats:

- **CSV** (Comma Separated Values) – To display data in spreadsheet format. A dynamic CSV view consists of a DDF file, created expressly for the view, and a TOK file.
- **XML** – To display data in XML format. A Dynamic XML view consists of a DDF file created expressly for the view.
- **Chart** – To present data in one of several chart formats. A dynamic chart view consists of a Properties file.
- **XSLT** – To convert statement data into XML; this enables you to present the data in a variety of formats, including CSV, HTML, WML, VXML, and QIF. An XSLT view consists of an XSLT file and an application DDF.
- **XML Query** – To access a view of dispute, annotation, and detail data from the database. The Detail Extractor job extracts this information from the data input file and loads it into the database. An XML view consists of a DDF file.
- **XS** – To place your application's XSL stylesheets in a live production environment (if you use XML data input files only). Create an XS type dynamic web view for each stylesheet (or set of alternate stylesheets) required to present a particular view of your XML data. eStatement Manager uses these stylesheets to identify and format the type of data the user requests on-demand.

When setting up a new eStatement application, you make the dynamic Web view files available to an application by "publishing" them using the eStatement Publisher tool. See "What Files Do I Need to Publish?" on page 19.

When publishing a dynamic view, you give it a name, such as AccountSummary, CallDetail, etc. You can use the same view names in multiple applications. Each would represent entirely different views, using different design files in each application. (You could also have multiple versions of the same application, for example, NatlWireless version 1.0 and NatlWireless version 2.0, which would have views with the same names.)

TIP: Publish application files in bulk to save time moving an application between development, testing, and production servers. See Command Center Help for information about bulk publishing.

Sample eStatement application

Your eStatement Manager installation includes the sample application called "NatlWireless," designed to simulate a telecommunication provider's application. NatlWireless uses a sample ASCII data file called *NatlWireless.txt*.

National Wireless sample files are located in the *\EDX_HOME\samples\NatlWireless* directory. *EDX_HOME* is the directory where you installed eStatement Manager (default is *eStatement*).

The Indexer job for National Wireless requires *Indexerjob\NatlWirelessIndexer.ddf*, which you would publish as part of the Indexer job configuration.

Sample dynamic HTML views

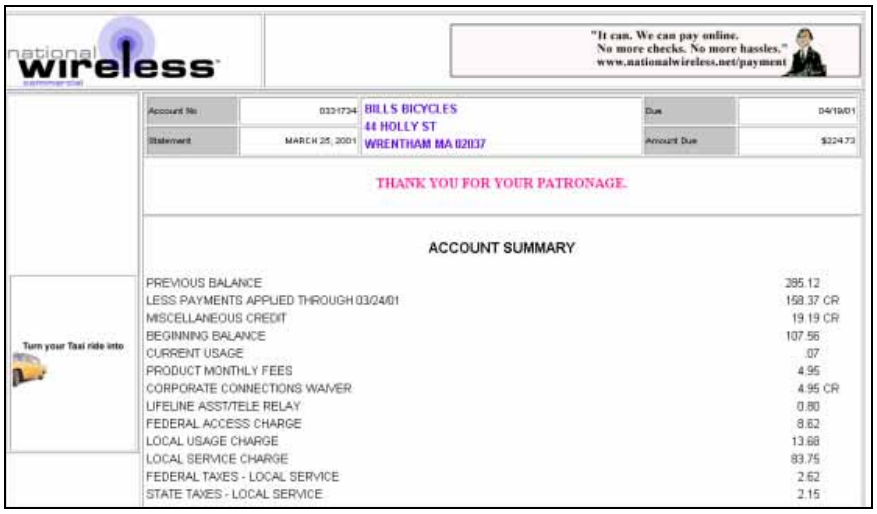
NatlWireless provides two dynamic HTML views to simulate dynamic statements, called HtmlDetail and NW_LocSummary.

HtmlDetail – Summary statement view – The main dynamic HTML view consists of the following files:

- *NatlWireless.DDF*

- NatlWireless.ALF
- NatlWireless.HTM

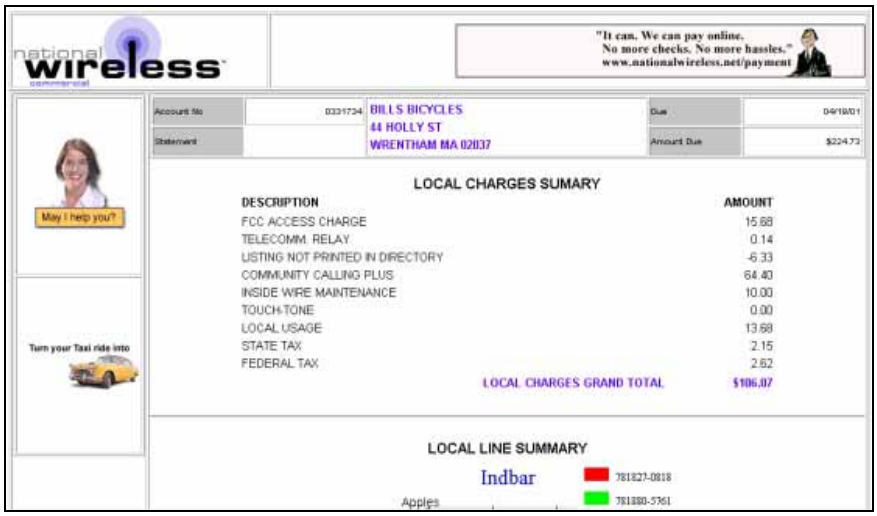
This view presents the primary page of the NatlWireless application:



NW_LocSummary – Detail statement view – A second dynamic HTML view used with the NatlWireless statement consists of the following files:

- *NW_LocSummary.DDF*
- NW_LocSummary.ALF
- NW_LocSummary.HTM

The composed view presents the following page of local call detail for NatlWireless:



Sample dynamic CSV, XML, XSLT, and XMLQuery views

In addition to the HTML view files, NatlWireless provides the following sample dynamic views:

Job Type	View Name	NatlWireless Sample Files
CSV	User-provided	*NatlWireless.DDF* *NatlWireless.TOK*
XML	User-provided	*NatlWireless.DDF*
XSLT	SummaryInfo	*NatlWireless.DDF* *XSLTDownload\summary_info_csv.XSL*
XMLQuery	AnnotationQuery DetailQuery DisputeQuery	*XMLQuery\annot_sql.XML* *XMLQuery\detail_sql.XML* *XMLQuery\dispute_sql.XML*

For a complete listing of National Wireless sample files, see the *Deploying and Customizing J2EE Applications Guide for Oracle Siebel eStatement Manager*.

Email Notification Views

eStatement Manager lets you communicate with users by creating and sending an email notification to enrolled users, typically to let them know that an online statement is available for viewing.

An email notification view consists of an ALF/DDF pair, plus one or more HTML files designed specifically to generate a particular email message.

To compose and send an email notification message, you use an EmailNotification job, which uses an associated email notification view along with the user's enrollment data, input data file, and index data from the database (generated by the Indexer job) to generate the email.

Example

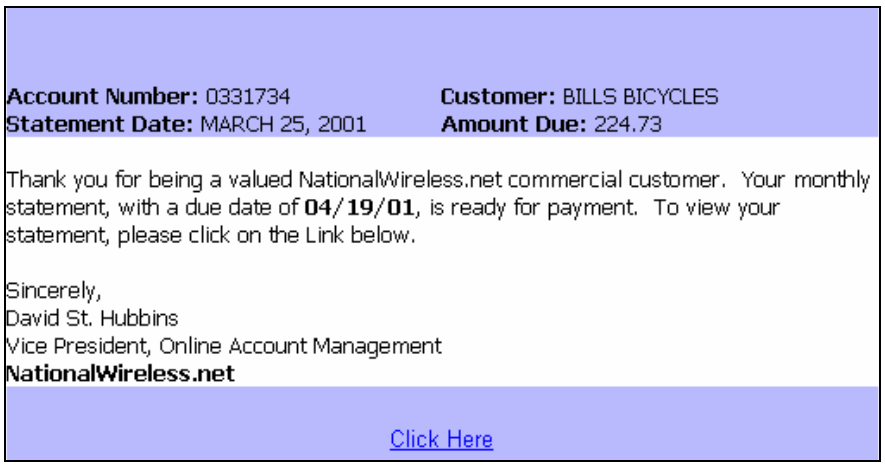
The NatlWireless sample application contains a sample email notification view, consisting of the following files, to generate a sample email for telecommunication customers:

- *NatlWireless.DDF*
- NW_Email.ALF
- NW_Email.HTM
- NW_EmailAlternate.HTM

(Note that this view also requires the index data created when you run the NatlWireless Indexer job using *Indexerjob\NatlWirelessIndexer.DDF*.)

It also contains an auxiliary HTML template called *NWEmailAlternate.HTM*. Conditional business logic in the ALF file determines which template eStatement Manager uses to generate the email notification.

A composed email notification using the *NW_Email.HTM* template looks like this:



HTML Output Views

If you generate static HTML-formatted output using an HTML Output job, your application uses a view containing an ALF/DDF pair and one or more corresponding HTML templates.

Detail Extractor Views

A Detail Extractor job extracts a single table or group of data and uploads it into a database table. The view files you publish to extract a single view of recurring table data includes a DDF file, a database table schema XML file, and a statement XSLT style sheet.

eStatement Manager provides the following sample files with the NatlWireless application to demonstrate publishing and using a view called "dtlextr" for the Detail Extractor job.

- *NatlWireless.DDF*
- DetailExtractor\summary_info.XML
- DetailExtractor\summary_info.XSL

What Files Do I Need to Publish?

Setting up a new application requires you to publish application design files. "Publishing" identifies the design files an application uses and lets you move the files from the design environment to your application server.

You must publish

- Certain files for each job configuration
- Your application's dynamic Web view files

You use Publisher, accessed from the Command Center Main Console, to publish design files. You can publish dynamic and batch job version sets one at a time or all together in bulk.

TIP: Publish application files in bulk to save time moving an application between development, testing, and production servers. See Command Center Help for information about bulk publishing.

For information about publishing dynamic Web views (version sets) individually, see "Publishing Your Application's Dynamic Web Views" on page 52.

Instructions for publishing batch job files individually are included with the individual procedures on creating and configuring each job type.

Files You Publish with Job Configurations

When you create and configure eStatement batch jobs, you publish the following files as part of (during) the job configurations:

Batch Job Type	Files you are instructed to publish as part of the job configuration:
Indexer	A DDF file for the Indexer task.
Email Notification	The email notification view: a DDF, an ALF, and one or more HTML files
HTML Output	A DDF file for the Indexer task, and the HTML Output view files: a DDF, ALF, and one or more HTML templates
XML Output	A DDF file (for use by both the Indexer and XMLFormatter tasks in this job)
Detail Extractor	A DDF file, database table schema XML file, and an XSLT style sheet for use by the StatementsToIR task
Report	An XML report description file and a DDF for dynamic XML
XML Email Notification	An XSL file (for XML input)
XML Loader	A customized XML schema file (.xsd) and the eStatement attribute file (edx.xsd)

Chapter 3 guides you through the process of creating and configuring jobs, and publishing the required design files.

Files You Publish as Dynamic Web Views (Version Sets)

After creating and configuring your application jobs, you publish the files that make up each dynamic Web view in your application.

"Publishing" a dynamic Web view identifies each file belonging to a particular view, and lets you name the view. It also lets you move the files from the design environment to your application server. When a user clicks a link to their statement ("running" a dynamic Web view), the link identifies the set of design files to use to extract and present the user's statement data. You can publish version sets one at a time as needed or publish an all version sets for an application in bulk.

Publishing adds a timestamp to the set of view files. A **version set** is a dated set of design files. Publishing a view is also called "**creating a version set**."

The following table shows the type of files that make up each type of dynamic Web view:

Dynamic Web view format	Files you publish in a version set for this view format:
HTML	DDF ALF One or more HTML files
XS	One or more XSL files
CSV	DDF TOK
XML	DDF
Chart	A Properties file
XSLT	XSL DDF
XML Query	XML

See "Publishing Your Application's Dynamic Web Views" on Page 52 for instructions.

For information about publishing in a live production environment, see Page 55.

Publishing a Readme.txt File with a Version Set

To help you identify and differentiate version sets, you can publish a file called readme.txt for each version set (for batch jobs or dynamic Web views) containing a description of the new or updated version set. When you search for version sets in Publisher, it displays the first line of the readme.txt along with the names of the files in the version set. You can include readme.txt files in bulk publishing as well. See Command Center Help for details.

3 Setting up a New Application and Jobs

Logging into the eStatement Manager Command Center

You use the Command Center to set up, configure, and manage your applications.

During live production, you use the Command Center to schedule and run production tasks, monitor system activity, and perform other system administration activities.

The Command Center is a secure application that requires you to log in with an administrator's ID and password. If you forget the Command Center password, contact your system administrator or the person who installed eStatement Manager.

Always log out of the Command Center after completing a session. By logging out, you help maintain the security of the eStatement production environment and minimize the chance an application or job can be accidentally corrupted or destroyed.

To change the administrator's password, see Command Center Help.

To log into the Command Center:

1. Verify that the Web server and the database server are both running.
2. Launch your Internet browser.
3. Enter the URL for the eStatement Command Center servlet configured when eStatement Manager was installed, such as `http://dusky:7001/eBilling`
4. On the Login Administrator page, enter the administrator's ID and password. The default ID is **`admin`** and the password is **`oracle.`**

   If you can't access the Login Administrator page or eStatement Manager does not recognize the ID and password, consult your system administrator or the person who installed eStatement Manager.
5. Click **`Submit`**. eStatement Manager displays the Command Center Main Console.

To log out of the Command Center:

- Click **`Logout`** on the Main Console.

Creating a New Application

To create a new application:

1. Go to the eStatement Manager Command Center.

   Click **`Create New Application`** from the Main Console. Command Center displays the Create New Application screen.

2. Enter the name of the application. The first character in the name must be an alpha. The rest of the name can contain alphanumerics and underscores, but no spaces.

3. Enter the JNDI name of the datasource EJB to use for this eStatement application. Use the real/global JNDI name as opposed to the local JNDI name ("java:comp/env/…"). The datasource EJB exists in a separate presentation EAR file. To successfully create the application, the JNDI name must exist and the EJB must be properly deployed and available to eStatement Manager. The Command Center validates the JNDI name before the mapping is persisted. For more details, see "About mapping your application to a datasource EJB" below.

4. From the Index Partition Count drop-down list, select the number of database partitions to use for the application index table. The number of tables you need is dependent on your database platform and the anticipated volume of data. For an Oracle database, we recommend you create one index table and use Oracle's native table partitioning functionality. For DB2 and SQLServer, we recommend using 4 or 12 index tables for quarterly or monthly index tables.

5. Click **`Create Application and Continue`**. Command Center displays the Create New Job screen.

6. Proceed with the instructions to create and configure an Indexer job for your application.

About mapping your application to a datasource EJB

You must specify a datasource EJB for each eStatement application (DDN) you create in the Command Center. When creating an eStatement application in the Command Center, a datasource refers to an EJB in your application (EAR file) that specifies summary information and location of your document data.

Specifying the datasource EJB at the DDN level allows you to set the JNDI mapping without modifying deployment descriptors, repackaging, and redeploying your web application. It also enables you to retrieve, for example, live data from an external database or archival data from offline storage. In some cases, customizing the datasource can also improve performance and save disk space.

Each web application shipped with eStatement Manager has a default datasource.

Web Application	Datasource Name (default EJB)
Sample	**`edx/Sample/ejb/EdocsDataSource`**
Training	**`edx/Training/ejb/EdocsDataSource`**

For information on developing a custom datasource EJB, please consult your Oracle Professional Services representative.

Creating and Configuring an Indexer Job

You must create and configure an Indexer job if you plan to use live retrieval with your application.

Creating and configuring an Indexer job requires you to:

- Specify configuration settings for the four production tasks that run sequentially as part of the Indexer job: Scanner, Indexer, IXLoader, and AutoIndexVolAccept.
- Publish the DDF file created expressly for your Indexer job.

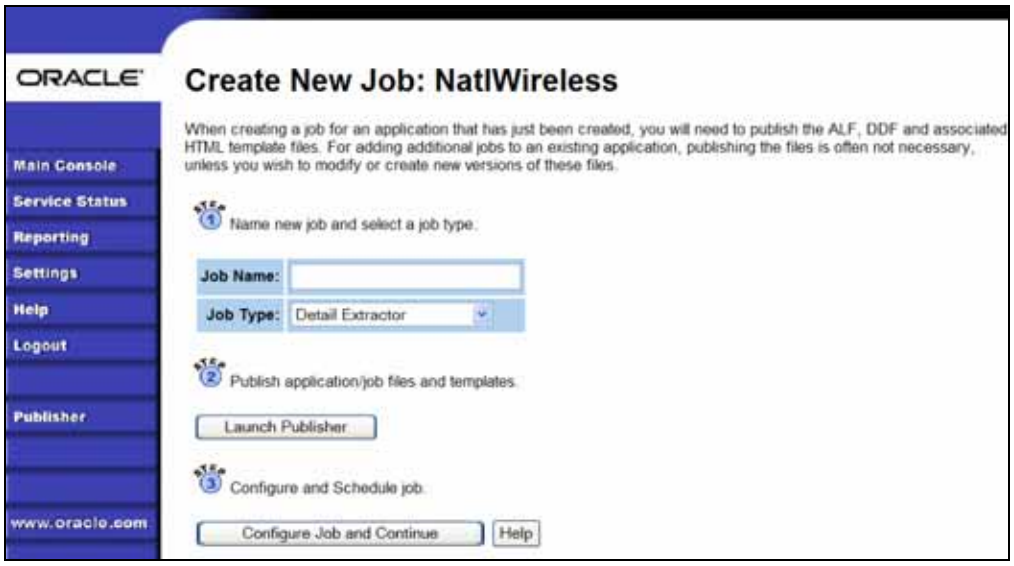
Review all the task and field configuration settings (in this section) to determine which options to use in your application.

Each time you run an Indexer job, it looks for multiple data files in the input directory and processes them one at a time.

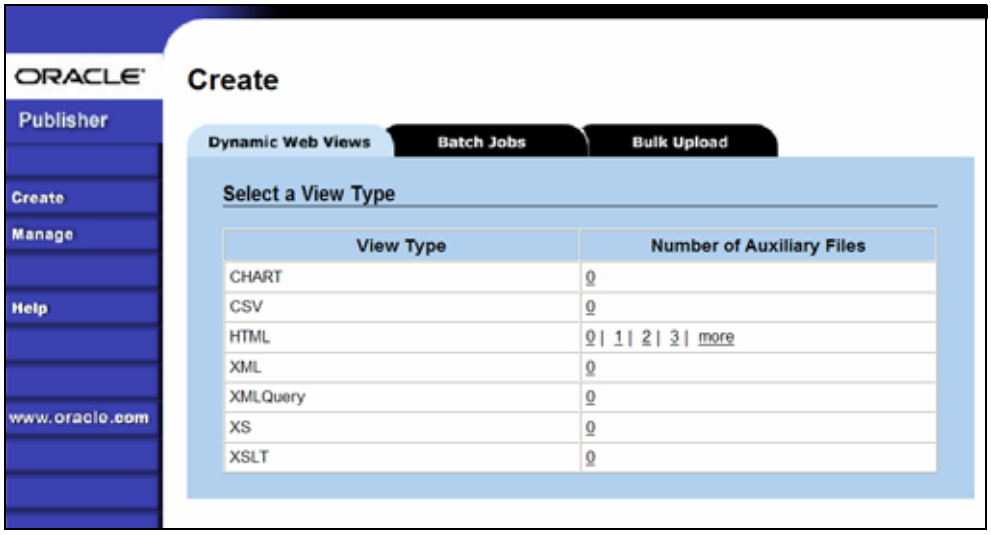
TIP: You can publish the Indexing DDF at the time you create and configure the Indexer job or you can publish it and all other required application files in bulk. Bulk publishing makes it easier to move application files between development, testing, and production servers. See Command Center Help for details.

To create and configure an Indexer job:

1. The Create New Job screen appears automatically after you create a new application. Otherwise click the application name on the Main Console, then click the **Add New Job** button. Command Center displays the Create New Job screen:

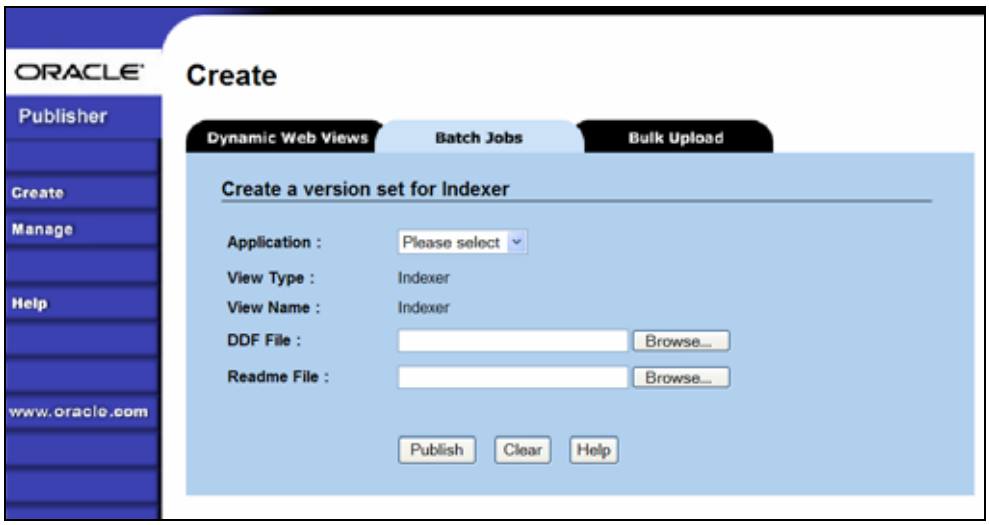


2. Enter a meaningful name for the Indexer job. The job name can contain alphanumerics and underscores, but no spaces. The Indexer job name cannot start with a numeric.
3. Select the Indexer job type from the drop-down menu.
4. If you plan to publish this application's files in bulk, skip to Step 13. Otherwise, click **Launch Publisher**. See Command Center Help for information about bulk publishing.
5. Click **Create** on the Publisher Menu.



6 Click the **Batch Jobs** tab.

7 For job type Indexer, click **0** (Number of Auxiliary files). Publisher displays the Create a Version Set For Indexer screen:



8 Select the application name from the drop-down list.

9 Browse and select the DDF file for the Indexer job.

10 Browse and select the readme.txt for this version set (if you've created one). See Command Center Help for details about the readme.txt.

11 Click **Publish**. If successful, Publisher displays the message "This Version Set has been Published Successfully" and displays details about the view files.

12 Close the Publisher window.

13 At the Create New Job screen in Command Center, click **Configure Job and Continue**. Command Center displays the job configuration screen:

14 Specify the configuration parameters for each of the four tasks that run as part of the Indexer job. Carefully read the descriptions of each task and field (in the section below) to choose the appropriate values for your application.

15 When finished entering configuration parameters, click **`Submit Changes and Schedule`**. Command Center asks "OK to submit this configuration?"

16 Click **`OK`**. Command Center submits the job configuration parameters and displays the Schedule screen. You can specify the Indexer job schedule later.

17 Click **`Main Console`**.

Task 1: Scanner

The Scanner task scans the input directory for new data input files. When it finds a new data file, it moves the file to the output directory and renames it, adding a timestamp (*YYYYMMDDHHMMSS_filename.ext*). If the Scanner finds multiple files, it processes them one at a time.

Scanner Task Input:	**Scanner Task Output:**
• Data source in input directory	• Data file moved to the data file path location • Timestamp added to the data file name

Scanner Task Configuration (Indexer job)	
Field	**What to enter**
Input File Path	Specify the input file data directory where Scanner can find the application's data input file. For UNIX systems, you must place the input file on the same file system as the data file. This file path can be on a NFS Mount. Only use a symbolic link on the same file system. The default is *eStatement/Input/Application Name* (where *eStatement* is your *EDX_HOME*, the default directory where you installed eStatement Manager, and *Application Name* is the name of your eStatement application).
Input File Name	Specify the name of your application's data input file. You can use wildcards (*) in the file name.
Output File Path	Specify the name of the application data directory where you want Scanner to move the data input file. The default is *eStatement /Data/Application Name* (where *eStatement* is your *EDX_HOME*, the default directory where you installed eStatement, and *Application Name* is the name of your eStatement application)

Task 2: Indexer

The Indexer task uses the data file from the data directory and the published Indexer DDF file and places index information for every data field into an XML Intermediate Representation (XIR) file.

Indexer Task Input:	**Indexer Task Output:**
• Data source file in the output file path • Most recent DDF published in AppProfiles directory (not configurable)	• XML Intermediate Representation (XIR) file

Indexer Task Configuration (Indexer job)	
Field	**What to enter/select**
DDF Path	(Not an editable field.) The directory path and name of the DDF file this Indexer job uses appear in this field. (This is the DDF you publish when configuring the Indexer job.)

Task 3: IXLoader

The IXLoader task converts the XIR file into an Intermediate Representation (IR) file, then uses the database loader to load data from the .IR file to the database using the script information in the .CMD file. It creates a row in the index table for each primary key. IXLoader also creates the .LOG, .CTL, and .CMD files.

IXLoader Task Input:	IXLoader Task Output:

• XIR file
• Settings from the job configuration
	• An Intermediate Representation file (.IR) file. The first row of the IR file contains the following header fields: Z_Primary_Key, Z_DocDate, Z_Doc_ID, plus the index field names. The IR file contains the following metadata:
	  ◦ The primary key
	  ◦ The date/time the file is processed, from the application server clock or a date specified as Z_DocDate from the data input file
	  ◦ A document ID, which uniquely identifies the statement
	  ◦ Byte offset information (position within the data file where customer's information is located)
	  ◦ The number of pages of a customer's statement
	  ◦ The index field list
	• Rows added to index tables
	• LOG, .CTL, and .CMD files

IXLoader Task Configuration (Indexer job)		
Field	**What to enter/select**	
Load Method	Choose one of the following database load methods:	
	Direct load	Stores data directly to the database. This option locks the index table, loads the information, and ends the call to the database. No sharing of tables is allowed during this process. A direct load is usually the fastest method. Oracle recommends method for Oracle and SQL Server.
	Conventional load	Uses Insert statements, one row at a time. Performs multiple selects and inserts on the table at once, but does not lock up the database and lock out Web users. A conventional load is usually slower than a direct load. Oracle recommends this method for DB2.

Task 4: AutoIndexVolAccept

The AutoIndexVolAccept task determines whether the system can make the Indexed data available for immediate user access or whether it must wait for you to approve the data. This task is primarily intended for eStatement applications using a customized verification process.

The verification process lets someone inside your organization see the data, but not customers. Using the internal verification process, you can mark the volume as approved, making it available for users. A volume is a data input file that has been successfully processed by the Indexer job and referenced in the volumes table.

AutoIndexVolAccept Task Input:	**AutoIndexVolAccept Task Output:**
• Settings from the Indexer job configuration	• If AutoAccept, fills in the Date Accepted field and the volumes table. • If Intercept to Verify, does not fill in a date field and it is up to your internal verification process to assess the date. The user cannot view their data until this process is complete

AutoIndexVolAccept Task Configuration (Indexer job)		
Field	**What to enter/select**	
Action on Index Volume	Choose one of the following options:	
	Auto Accept	(Default) Choose this option if you are not using a custom verification/audit application. Auto Accept automatically makes the index references immediately available (viewable) to the user. Auto Accept fills in the Date Accepted field in the volumes table. A date in this field means the volume has been approved.
	Intercept to Verify	Choose this option if you use an internal verification process, which lets you perform quality control on the data then accept or reject it. You can accept or reject the entire set of data, but not individual documents. Users cannot access this data until the verification process is complete.

Creating and Configuring an EmailNotification Job

You create an EmailNotification job to create and send an email notification message to enrolled users.

Creating and configuring an EmailNotification job requires you to:

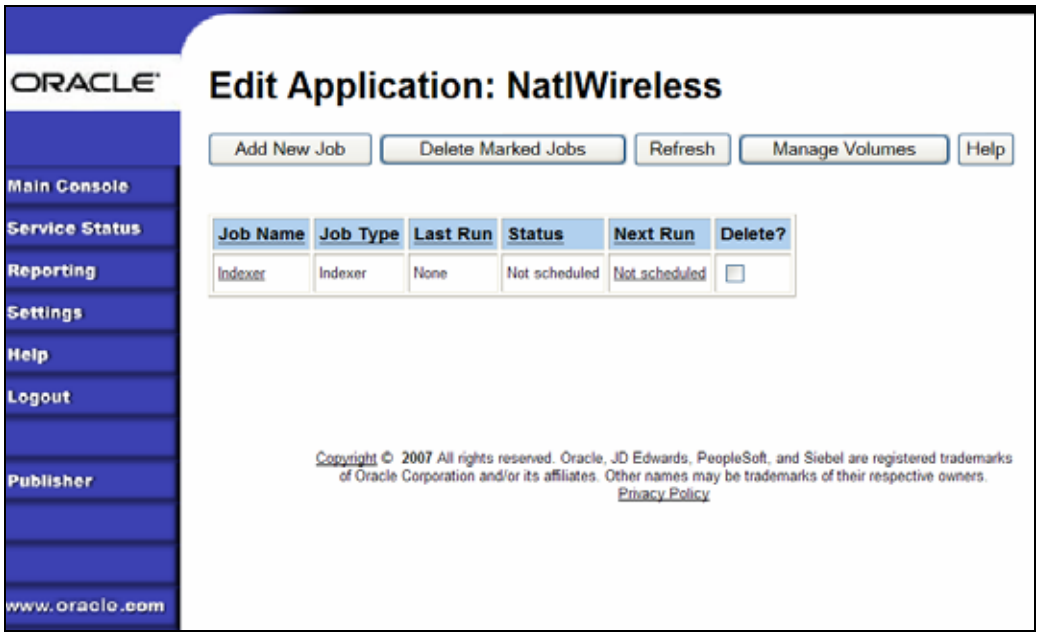
- Specify configuration settings for the two production tasks that run sequentially as part of the job: IVNScanner and MailNotification
- Publish the email view files (DDF, ALF, and HTML templates) created for the intended email message

Review all the task and field configuration settings in this section to determine which options to use in your application.

Tip: You can publish the EmailNotification view files (DDF, ALF, and HTML's) at the time you create and configure the EmailNotification job or you can publish these and all other required application files in bulk. Bulk publishing makes it easier to move application files between development, testing, and production servers. See Command Center Help for details.

To create and configure an EmailNotification job:

1 On the Main Console, click the application name in the table. The Edit Application screen appears.



2 Click **Add New Job**. Command Center displays the Create New Job screen.

3 Enter a meaningful name for the job. The job name must start with an alpha character. The rest of the characters can be alphanumeric and can contain underscores, but no spaces.

4 Select the Email Notification job type from the drop-down menu.

5 If you plan to publish this application's files in bulk, skip to Step 13. Otherwise click **Launch Publisher**. See Command Center Help for information about bulk publishing.

6 Click **Create** on the Publisher Menu.

7 Click the **Batch Jobs** tab.

8 For job type Email Notification, click the number of auxiliary HTML templates the view uses. (The number of HTML templates *in addition to* the default template. If the view uses one HTML file, click **0**, if it uses two HTML templates, click **1**, etc.) Publisher displays the Create a Version Set for Email Notification screen:

9 Select the application name from the drop-down list. Browse and select the DDF, ALF, and HTML files in the version set.

10 Browse and select the readme.txt for this version set (if you've created one). See Command Center Help for details about the readme.txt.

11 Click **Publish**. If successful, Publisher displays the message "This Version Set has been Published Successfully" and displays details about the view files.

12 Close the Publisher window.

13 At the Create New Job screen in Command Center, click **Configure Job and Continue**. Command Center displays the EmailNotification job configuration screen.

14 Specify the configuration parameters for each EmailNotification task. Carefully read the descriptions of each task and field to choose the values appropriate for your application and job.

15 When finished entering configuration parameters, click **Submit Changes and Schedule**. Command Center submits the job configuration parameters and displays the Schedule screen. You can schedule the EmailNotification job later; see Command Center Help.

16 Click **Main Console**.

Task 1: IVNScanner

The IVNScanner task determines whether index data has been verified before creating and sending email to enrolled customers. This task is primarily intended for applications with a customized verification/audit application.

IVNScanner looks for a date processed in the Date Accepted (or Date Rejected) column in the volumes table. A volume is a data input file that has been successfully processed by the Indexer job and referenced in the volumes table. The Indexer job must run before EmailNotification. The EmailNotification job processes one indexed volume at a time until IVNScanner finds no more newly indexed volumes listed in the volumes table.

If you selected "Intercept to Verify" in the Action on Index Volume option when configuring the AutoIndexVolAccept task in the Indexer job, you must use your customized verification/audit application to either accept or reject the indexed data before the EmailNotification job can process email for that data. (You can optionally choose to send email when the volume is rejected; this is useful in a test environment only.)

IVNScanner Task Input:	**IVNScanner Task Output:**
• A date processed value in the Date Accepted or the Date Rejected columns in the volumes table	• (None)

IVNScanner Task Configuration (MailNotification job)		
Field	**What to enter/select**	
Index Volume Status	Specifies that the job can proceed to create and send email to customers when a date appears in either the Date Accepted or Date Rejected column in the volumes table.	
	Accepted	(Default) Choose this option if you do not have custom verification/audit application or if you have one and you want to send out email only after a volume has been approved. If you choose this option, IVNScanner looks for a date in the Date Accepted column; if it contains a date, it proceeds to generate email.
	Rejected	Use this setting to send email if a volume has been rejected in the custom verification application (use this option for testing purposes only). If you choose this option, IVNScanner looks for a date in the Date Rejected column; if it finds a date, it proceeds to generate email.
Scan Starting From (Number of Days)		Specify how many previous days' volumes to scan for; IVNScanner selects any volumes indexed on or between the current date and the number of days ago you specify.

Task 2: MailNotification

The MailNotification task builds and sends email notifications. It uses information from the job configuration settings, the most recently published email version set, index references, the data file, and the email addresses to generate email.

The configuration settings you must specify for this task include the Base URL, the mail server name, the return address, the administrator's address, subject lines, etc., and how eStatement Manager handles delivery errors.

You also specify enrollment model settings to tell eStatement Manager where to look for email enrollment information. If you use eStatement Manager enrollment, the system uses the email addresses found in the eStatement database by default. If you use a customized enrollment application, you must specify the JNDI name of your custom account resolver.

MailNotification Task Input:	**MailNotification Task Output:**
• Index data • Data file • Email version set (.DDF, .ALF, .HTML, auxiliary files) • Task configuration settings • Recipient email information	• Emails sent to the specified SMTP for delivery • Flag added to the database indicating "sent" status flag for reporting purposes

MailNotification Task Configuration	
Field	**What to enter/select**
Base URL	Specify the Web address where the user can view their statement after receiving an email (this is usually included as a link in the email message).
SMTP Hosts	Specify the mail server name. You can specify multiple host systems, separated by commas.
Return Address	Specify the "from" address that appears in the notification. The user can reply to this address.
Subject Text	Enter the text to appear in the subject line in email notifications.
Administrator's Addresses	Specify the address to receive email if there's a problem passing it to the SMTP host or if it is not working properly for some other reason. Enter multiple addresses separated by commas (the mail goes to all addresses).
Administrator Subject Text	Enter the text to use in the subject line of messages sent to the administrator addresses.
Max Number of Retries	Specify the number of times eStatement Manager should try sending the mail to the mail server in the event of an error (the server could be busy or down) before the job stops.
Retry Period (min)	Specify how frequently eStatement Manager should retry sending email if it fails. The default is 60 minutes.

MailNotification Task Configuration	
Account Resolver	Specify the JNDI name of the Account Resolver your application uses.
Access Type	This field provides a way to pass an extra parameter to one of Oracle enrollment applications (CDA or Any other enrollment application). Leave this field blank if you do not have customizations to either Oracle application, or if you use a customized enrollment application. Consult the programmer if your application requires passing an extra parameter.
Auditor Model	Specify the JNDI name of the Mail Auditor your system uses. Contact your system administrator to determine whether custom mail auditing has been implemented and if so, to obtain the appropriate JNDI name of your mail auditing client.

Creating and Configuring a MessageFailRecovery Job

The MessageFailRecovery job enables you to recover the failed email messages that are sent by the job alerts and payment emailing features. If the email delivery fails because the mail server goes down or because of other network issues, running the MessageFailRecovery job will send the unsent email.

MessageFailRecovery does not correct problems with email unsent due to bad email addresses.

There are no configuration parameters for this job.

To create and configure a MessageFailRecovery job:

1. On the Main Console, click the application name in the table. The Edit Application screen appears.
2. Click **Add New Job**. Command Center displays the Create New Job screen.
3. Enter a meaningful name for the job. The job name must start with an alpha character. The rest of the characters can be alphanumeric and can contain underscores, but no spaces.
4. Select the **MessageFailRecovery** job type from the drop-down menu.
5. You do not need to publish files with this job. Click **Configure Job and Continue**.
6. Click **Submit Changes and Schedule**. Click **OK**.
7. Enter a **Start Date** and a **Start Time**. Do not schedule this job; run it only as needed.
8. Click **Save Schedule**. Command Center submits the job configuration.

Creating and Configuring a Purge App Job

Create and configure the Purge App job to periodically remove index, email, reporting, and PWC data, and detail database tables (if any) from your application database tables. Running this job frees up space on your database server.

You also need to run a Purge App job to eliminate index references to historical statement data to prevent it from being included in summary page information.

Purge App also purges any sub-document index data when purging the data for its parent, or root document.

CAUTION: Once you run a Purge App job, the data deleted will no longer be available for statistical reporting.

When you configure a Purge App job, you specify settings for the four production tasks that run sequentially as part of the job:

- Task 1: PurgeIndexData – Purges records from the index, detail, annotations, and dispute tables. You can specify separate purge criteria for rejected and accepted IVN data. PurgeIndexData also lets you purge indexed source files (.AFP, txt, etc.) as well as auxiliary .XIR files generated during indexing based on your selection criteria. You also have the option to save index data to an archive file before purging.
- Task 2: PurgeEmailData – Purges information from the mail queue table. This table grows rapidly if you run Email Notification jobs.
- Task 3: PurgeActivityData – Purges user and system activity data.
- Task 4: PurgePWCData – Purges job and task instance data. These tables grow rapidly if you run statement-based processing jobs, such as the Report job.

Review the task and field configuration settings in this section to determine which options to set for your application. Note that you do not need to publish files for a Purge App job.

To create and configure a Purge App job:

1. On the Main Console, click the application name, listed under Applications in the table. The Edit Application screen appears.
2. Click **Add New Job**. Command Center displays the Create New Job screen.
3. Enter a meaningful name for the Purge App job. The job name must start with an alpha character. The rest of the characters can be alphanumeric and can contain underscores, but no spaces.
4. Select job type Purge App. (You do not publish files for a Purge App job.)
5. Click **Configure Job and Continue**. Command Center displays the Purge App job configuration screen.
6. For each task, specify the configuration parameters appropriate for your application and database, including whether to purge the particular type of data and the number of prior days' data to purge. To skip a task, uncheck the box next to "Purge datatype" in that task's configuration. For each task you want to run, specify the age of the data you want to purge. Select **All** to purge all data up to midnight of the current date, or specify a number of days, which purges all data up to

midnight of the (relative) day specified. The limit is 3650 days, or ten years. See the next section for a description of each task and additional configuration parameters available.

7 When finished selecting configuration parameters, click **`Submit Changes and Schedule`**. Command Center submits the job configuration parameters and displays the Schedule screen. You can schedule the Purge App job later.

8 Click **`Main Console`**.

Task 1: PurgeIndexData

PurgeIndexData purges records from the index, detail, annotation, and dispute tables with a date processed outside the range you specify. You can specify separate purge criteria for rejected and accepted IVN data.

This task also provides the options to purge indexed source files (.AFP, .txt, etc.) and auxiliary .XIR files generated during indexing based on your selection criteria, and to save index data to an archive file. Note that if you do not purge data input files, they remain in the data directory and you must manually remove them as needed.

For each document, PurgeIndexData purges the index references, detail tables (loaded by the Detail Extractor job), and the related dispute and annotations information, regardless of their state or the date the information was submitted.

You have the option to skip the PurgeIndexData task with the Purge App job.

PurgeIndexData Task Configuration	
Field	**What to enter/select**
Purge Rejected Data	Check to include rejected index data in the purge; uncheck to skip.
…And files	Check to include rejected data input files (.AFP, .TXT, etc) and associated auxiliary .XIR files in the purge; uncheck to skip.
All	Select to purge all rejected index data (and optionally, files) with a date processed up to midnight of the current date.
Older than (number of days)	Select to specify the number of days older than which rejected index data is included in the purge. Purge App purges all rejected index data with a processed date up to midnight of the (relative) day specified. The limit is 3650 days, or ten years.
Purge Accepted Data	Check to include accepted index data in the purge; uncheck to skip.
…And files	Check to include accepted data input files (.AFP, .TXT, etc) and associated auxiliary .XIR files in the purge; uncheck to skip.
All	Select to purge all accepted index data (and optionally, files) with a date processed up to midnight of the current date.
Older than (number of days)	Select to specify the number of days older than which accepted index data is included in the purge. Purge App purges all accepted index data with a date processed up to midnight of the (relative) day specified. The limit is 3650 days, or ten years.

PurgeIndexData Task Configuration	
Field	**What to enter/select**
Purge by Partition	Check to use the Purge By Partition feature. This feature is Oracle database-specific and is not implemented for other database platforms. See "Using the Purge by Partition Feature" on page 39.
…And files	Check to include accepted data input files (.AFP, .TXT, etc) and associated auxiliary .XIR files in the purge; uncheck to skip.
Older than (number of days)	Select to specify the number of days older than which all data is included in the purge. Purge App purges all data with a date processed up to midnight of the (relative) day specified. The limit is 3650 days, or ten years.
Archive to (subfolder)	Check this box to store the index data and (optionally) files to an archive file before purging. Purge App creates a .zip file under EDX_HOME\PurgeArchiveData\DDN or a directory you specify. The name of the .zip file is the timestamp of when the purge occurred in format YYYYMMDDhhmmss.zip. The .zip file contains a directory for each volume purged, such as IVN-1. These directories store .dat files containing the deleted rows from the index, disputes, and annotations tables. The .zip file also contains data input and auxiliary files (.AFP, .XIR, CTL, logs, etc.) from the Indexer job if selected for purge in the configuration; see previous configuration options.

Task 2: PurgeEmailData

Purges records from the dynamic mail queue table with a date processed that falls outside the age you specify here. This table can grow rapidly if you send email.

You have the option not to purge email data with the Purge App job.

PurgeEmailData Task Configuration	
Field	**What to enter/select**
Purge Email Data	Check to include email data in the purge; uncheck to skip this task.
All	Select to purge all email data with a date processed up to midnight of the current date.
Older than (number of days)	Select if you want to specify the number of days older than which email data is included in the purge. Purge App purges all email data with a date processed up to midnight of the (relative) day specified. The limit is 3650 days, or ten years.

Task 3: PurgeActivityData

PurgeActivityData purges user and system activity records from the user activity table with a date processed that falls outside the age you specify.

You have the option not to purge report data when you run the Purge App job.

PurgeActivityData Task Configuration	
Field	**What to enter/select**
Purge Activity Data	Check to include activity data in the purge; uncheck to skip this task.
All	Select to purge all activity data with a date processed up to midnight of the current date.
Older than (number of days)	Select if you want to specify the number of days older than which activity data is included in the purge. Purge App purges all activity data with a date processed up to midnight of the (relative) day specified. The limit is 3650 days, or ten years.

Task 4: PurgePWCData

PurgePWCData purges PWC (job and task instance) data from the database with a date processed that falls outside the age you specify. If you are running report jobs, these files can grow rapidly.

You have the option not to purge PWC data when you run the Purge App job.

PurgePWCData Task Configuration	
Field	**What to enter/select**
Purge PWC Data	Check to include PWC data in the purge; uncheck to skip this task.
All	Select to purge all PWC data with a date processed up to midnight of the current date.
Older than (number of days)	Select if you want to specify the number of days older than which PWC data is included in the purge. Purge App purges all PWC data with a date processed up to midnight of the (relative) day specified. The limit is 3650 days, or ten years.

Using the Purge by Partition Feature

The Purge by Partition feature lets a database administrator change the Indexer table to implement an Oracle native partitioning feature to create boundaries on the table. Oracle recommends creating one Indexer table for clients with an Oracle database and using Oracle's native table partitioning to partition the Indexer table.

The Purge by Partition feature is Oracle database-specific and is not implemented for other database platforms.

To use this feature, the *edx_partition* package must exist on the Oracle database server.

To use the Purge by Partition feature:

1. Create an application with a partition count of one (1).
2. Run the Indexer job and make sure the indexed data loaded properly.
3. Create a partition table with a structure equal to the Indexer table (I_24_Y1), but with a different name, such as temp_I_24_Y1, for example:

```
CREATE TABLE temp_I_24_Y1 (
  Z_PRIMARY           VARCHAR2 (255)  NOT NULL,
  Z_DOC_DATE          DATE          NOT NULL,
  Z_DOC_ID            VARCHAR2 (255)  NOT NULL,
  Z_IVN               NUMBER (9)    NOT NULL,
  Z_CONTEXT           VARCHAR2 (255),
  "CurrentCharges"    VARCHAR2 (13),
  "PymtTxt"           VARCHAR2 (24),
  "CustName"          VARCHAR2 (30),
  "StatementDate"     VARCHAR2 (20),
  "LateFee"           VARCHAR2 (5),
  "AmountDue"         VARCHAR2 (12),
  "FirstStmtIndicator"  VARCHAR2 (8),
  "LastStmtIndicator"  VARCHAR2 (10),
  "EastState"         VARCHAR2 (4),
  "CentState"         VARCHAR2 (4),
  "WestState"         VARCHAR2 (4),
  "CallForward"       VARCHAR2 (12),
  "CallID"            VARCHAR2 (10),
  "CallWait"          VARCHAR2 (12),
  "VoiceMail"         VARCHAR2 (10),
  "CustType"          VARCHAR2 (3),
  Y_1                 VARCHAR2 (64),
  Y_2                 VARCHAR2 (64),
  Y_3                 VARCHAR2 (64),
  Y_4                 VARCHAR2 (64),
  Y_5                 VARCHAR2 (64),
  Y_6                 VARCHAR2 (64),
  Y_7                 VARCHAR2 (64),
  Y_8                 VARCHAR2 (64),
  Y_9                 VARCHAR2 (64),
  Y_10                VARCHAR2 (64),
  CONSTRAINT temp_I_24_Y1_PK
  PRIMARY KEY ( Z_DOC_ID, Z_DOC_DATE )
    USING INDEX TABLESPACE edx_app_data_idx  local
  )
  PARTITION BY RANGE (Z_DOC_DATE)
 (
 PARTITION p0 VALUES LESS THAN (to_date('11/01/2006', 'mm/dd/yyyy'))
 )  TABLESPACE edx_app_data;
```

In this example, the partition key is `z_doc_date` and table partitions were done on monthly basis.

4 Create indexes on the partitioned table (**temp_I_24_Y1**), for example:

```
CREATE INDEX temp_I_24_Y1_I1 ON temp_I_24_Y1(Z_PRIMARY)
  TABLESPACE edx_app_data_idx local;

CREATE INDEX temp_I_24_Y1_I2 ON temp_I_24_Y1(Z_IVN)
  TABLESPACE edx_app_data_idx local;

CREATE INDEX temp_I_24_Y1_I3 ON  temp_I_24_Y1(Z_DOC_DATE)
  TABLESPACE edx_app_data_idx local;

CREATE INDEX temp_I_24_Y1_I4 ON   temp_I_24_Y1(Z_CONTEXT)
  TABLESPACE edx_app_data_idx local;
```

5 Exchange the indexer table's (**I_24_Y1**) data into first partition in the partition table, for example:

```
ALTER TABLE temp_I_24_Y1  EXCHANGE PARTITION p0  WITH TABLE I_24_Y1
WITHOUT VALIDATION;
```

6 Rebuild the index of primary key column on partition p0, for example:

```
ALTER INDEX temp_I_24_Y1_PK REBUILD PARTITION p0;
```

7 Add a couple of partitions to the partitioned table, for example:

```
ALTER TABLE temp_I_24_Y1 ADD PARTITION p1 VALUES LESS THAN
(to_date('12/01/2006', 'mm/dd/yyyy'))
TABLESPACE edx_app_data;

ALTER TABLE I_24_Y1 ADD PARTITION p3 VALUES LESS THAN
(to_date('01/01/2007', 'mm/dd/yyyy'))
TABLESPACE edx_app_data;
```

8 Rename or drop existing indexer (**I_24_Y1**) table, for example:

```
ALTER TABLE I_24_Y1 RENAME TO old_I_24_Y1;
```

9 Rename partitioned table into original indexer table, for example:

```
ALTER TABLE temp_I_24_Y1 RENAME TO I_24_Y1;
```

10 Run the Indexer job for a few days.

11 Run the Purge App job with the "Purge by Partition" option selected to drop the partitions for the given date range.

12 Check the database to see whether the partitions were dropped properly. Log into SQL* PLUS as a database user and run the following command, using the actual table name:

```
Sql > select * from user_tab_partitions where table_name='I_24_Y1';
```

Creating and Configuring a Purge Logs Job

Create and configure the Purge Logs job to periodically remove historical information from the log table in the system database. Purge Logs removes data for all applications you have. Running this job frees up space on your database server.

When you configure a Purge Logs job, you specify settings for the Purge Logs production task. Note that you do not need to publish files for a Purge Logs job.

CAUTION: Purge Logs removes data globally (for all applications you have). Once you run a Purge Log job, the data deleted will no longer be available for inclusion on View Log reports.

To create and configure a Purge Logs job:

1. On the Main Console, click the application name, listed under Applications in the table. The Edit Application screen appears.
2. Click **Add New Job**. Command Center displays the Create New Job screen.
3. Enter a meaningful name for the Purge Logs job. The job name must start with an alpha character. The rest of the characters can be alphanumeric and can contain underscores, but no spaces.
4. Select job type Purge Logs.
5. Click **Configure Job and Continue**. Command Center displays the Purge Logs configuration screen:
6. Specify the configuration parameters for the PurgeLogs task.
7. When finished, click **Submit Changes and Schedule**. Command Center submits the job configuration parameters and displays the Schedule screen. You can schedule the Purge Logs job later.
8. Click **Main Console**.

PurgeLogs Task

The PurgeLogs task purges records from the logs table. It removes records with a date processed outside the range you specify here.

PurgeLogs Task Configuration		
Field	**What to enter/select**	
Purge Prior to (Number of Days)	Specify a value:	
	0	Purges all information to date
	1	Purges all information up to midnight of the previous day
	>1	Purges all information up to midnight of the (relative) day specified

Creating and Configuring an HTML Output Job

An HTML Output job creates a static HTML output file for each primary key. Configure an HTML Output job only if you plan to limit access to your database and let users view a static HTML output file only. Static HTML output files may be necessary if you partner with thin or thick consolidators for statement presentment.

CAUTION: An HTML Output job is necessary only if you plan to provide users access to static HTML output. Use an Indexer job with dynamic HTML Web views to provide live retrieval of HTML-formatted statements.

Creating and configuring an HTML Output job requires you to:

- Specify configuration settings for the three production tasks that run sequentially as part of the Indexer job: Scanner, Indexer, and HTMLFormatter.
- Publish the DDF file for the application's Indexer task (if you don't already have an Indexer job defined for the application).
- Publish the HTML Output view files (DDF, ALF, and HTML templates) defined for the application.

Review the task and field configuration settings in this section to determine which options to set for your application.

Tip: You can publish the HTML Output view files (DDF, ALF, and HTML's) at the time you create and configure the HTML Output job or you can publish these and all other required application files in bulk. Bulk publishing makes it easier to move application files between development, testing, and production servers. See Command Center Help for details.

To create and configure an HTML Output job:

1. On the Main Console, click the application name, listed under Applications in the table. The Edit Application screen appears.
2. Click **Add New Job**. Command Center displays the Create New Job screen.
3. Enter a meaningful name for the HTML Output job. The job name must start with an alpha character. The rest of the name can be alphanumeric and can have underscores, but no spaces.

4 Select job type HTML Output.

5 If you plan to publish this application's files in bulk, skip to Step 17. Otherwise click **Launch Publisher**. See Command Center Help for information about bulk publishing.

6 Click **Create** on the Publisher Menu.

7 Click the **Batch Jobs** tab.

8 For job type HTML Output, click the number of auxiliary HTML templates the view uses (the number *in addition to* the default template). If the view uses one HTML file, click **0**, if it uses two HTML templates, click **1**, etc.) Publisher displays the Create a Version Set for HTML Output screen:

9 Select the application name from the drop-down list.

10 Select the DDF, ALF, and HTML files in the HTML Output version set.

11 Browse and select the readme.txt for this version set (if you've created one). See Command Center Help for details about the readme.txt.

12 Click **Publish**. Publisher lets you know when it has successfully published the version set and displays details about the view files:

13 If you *don't* already have an Indexer job configured for this application, click the **Batch Jobs** tab and publish a DDF for the Indexer task that runs as part of the HTML Output job (If you already have an Indexer job for this application, go to Step 17.) Command Center displays the Create a Version Set for Indexer screen:

14 Specify the DDF file for the Indexer task.

15 Click **Publish**. Command Center displays the Submission screen for the DDF:

16 Close the Publisher window.

17 At the Create New Job screen in Command Center, click **Configure Job and Continue**. Command Center displays the HTMLOutput job configuration screen.

18 Specify the configuration parameters for each of the three tasks that run as part of the HTML Output job. Carefully read the descriptions of each task and field (below) to choose the values appropriate for your application and job.

19 When finished entering configuration parameters, click **Submit Changes and Schedule**. Command Center submits the job configuration parameters and displays the Schedule screen. You can specify the HTML Output job schedule later.

20 Click **Main Console**.

Task 1: Scanner

See page 27 for a description of Scanner task parameters.

Task 2: Indexer

See page 28 for a description of Indexer task parameters.

Task 3: StaticHtmlFormatter

The StaticHtmlFormatter task uses the data file and most recent HTML version set published (DDF, ALF, and HTML's) to generate an HTML-formatted static output file for each primary key in the data file.

StaticHtmlFormatter Task Input:	StaticHtmlFormatter Task Output:
• Data input file • Most recent HTML version set published (.DDF, .ALF, & HTMLs)	• A static HTML file for each primary key in the */Output/Application Name* directory

StaticHtmlFormatter Task Configuration	
Field	**What to enter/select**
DDF Path	(Not an editable field.) Displays the full pathname of the DDF published with the version set for this job.
ALF Path	(Not an editable field.) Displays the full pathname to the ALF file you published with the version set for the job.
HTML Template Path	(Not an editable field.) Displays the full pathname to the HTML template you published with the version set for this job.
Output Path	Enter the full pathname where you want eStatement Manager to place the Static HTML-formatted output. The default directory is *eStatement/Output/Application Name* (where *eStatement* is your *EDX_HOME*, the default directory where you installed eStatement Manager, and *Application Name* is the name of your eStatement application).

Creating and Configuring an XML Output Job

An XML Output job creates a static XML output file. Configure an XML Output job if you plan to generate a static XML output file for loading into another database.

CAUTION: An XML Output job is necessary only if you plan to generate static XML output. Use an Indexer job with dynamic XML Web views to provide users with live retrieval of XML-formatted statements.

Creating and configuring an XML Output job requires you to:

- Specify configuration settings for the three production tasks that run sequentially as part of the Indexer job: Scanner, Indexer, and XMLFormatter.
- Publish a DDF file for XML output (if you don't already have an Indexer job defined for this application.)
- Publish the DDF file for use by the Indexer task.

Review the task and field configuration settings in this section to determine which options to set for your application.

You must publish the DDF file created for your XML Output job.

Tip: You can publish the required DDF file at the time you create and configure the XML Output job or you can publish it and all other required application files in bulk. Bulk publishing makes it easier to move application files between development, testing, and production servers. See Command Center Help for details.

To create and configure an XML Output job:

1. On the Main Console, click the application name, listed under Applications in the table. The Edit Application screen appears.
2. Click **Add New Job**.
3. Enter a meaningful name for the XML Output job. The job name must start with an alpha character. The rest of the name can be alphanumeric and can have underscores, but no spaces.
4. Select job type **XML Output**.
5. If you plan to publish this application's files in bulk, skip to Step 16. Otherwise click **Launch Publisher**. See Command Center Help for information about bulk publishing.
6. Click **Create** on the Publisher Menu.
7. Click the **Batch Jobs** tab.
8. For the job type XML Output, click **0** (Number of Auxiliary files). Publisher displays the Create a Version Set for XML Output screen.
9. Select the application from the drop-down list.
10. Select the DDF file for the XML Output job.
11. Browse and select the readme.txt for this version set (if you've created one). See Command Center Help for details about the readme.txt.
12. Click **Publish**. Publisher lets you know when it has successfully published the version set and displays details about the DDF file.
13. If you *don't* already have an Indexer job configured for this application, click the **Batch Jobs** tab and publish a DDF file for the Indexer task that runs as part of the XML Output job. (If you already have an Indexer job for this application, go to Step 16.) eStatement Manager displays the Create a Version Set for Indexer screen.
14. Select the DDF file to use for the Indexer task and click **Publish**. eStatement Manager displays the DDF for Indexer task Submission screen.
15. Close Publisher.
16. At the Create New Job screen in Command Center, click **Configure Job and Continue**. eStatement Manager displays the XMLOutput job configuration screen.
17. Specify the configuration parameters for each of the three tasks that run as part of the XML Output job. Carefully read the descriptions of each task and field (below) to choose the values appropriate for your application and job.

18 When finished entering configuration parameters, click **`Submit Changes and Schedule`**. EStatement Manager submits the job configuration parameters and displays the Schedule screen. You can specify the XML Output job schedule later.

19 Click **`Main Console`**.

Task 1: Scanner

See page 27 for a description of Scanner task parameters.

Task 2: Indexer

See page 28 for a description of Indexer task parameters.

Task 3: XMLFormatter

The XMLFormatter task uses the data file and the DDF published with the version set for the XML Output job to generate a static, XML-formatted output file for each primary key in the data file.

XMLFormatter Task Input:	**XMLFormatter Task Output:**
• Data input file • Most recent DDF published for the XML Output job	• A static XML file in the *eStatement/Output/Application Name* (where *eStatement* is your *EDX_HOME*, the default directory where you installed eStatement Manager, and *Application Name* is the name of your eStatement application).

XMLFormatter Task Configuration	
Field	**What to enter/select**
DDF Path	Enter the full pathname to the DDF File. You must specify the same DDF file that you publish for the XML Output job Indexer task.
Output Path	Enter the full pathname where you want eStatement Manager to place the Static XML-formatted output. The default directory is *eStatement/Output/Application Name* (where *eStatement* is your *EDX_HOME*, the default directory where you installed eStatement Manager, and *Application Name* is the name of your eStatement application).

Creating and Configuring a Detail Extractor Job

A Detail Extractor job lets you to use XSL to upload recurring detail data from your input file to the eStatement Manager database. You can then use the uploaded data any way you want. For example, you could upload the contents of telephone call detail, merge it into your online statements, and let your Web users track line item disputes, add annotations, etc. You must write separate applications to retrieve the data and perform any functionality with the data.

You must configure and run a separate Detail Extractor job for each set of data (table or group of tables) you want to upload to a database table.

Creating and configuring a Detail Extractor job requires you to:

- Specify configuration settings for the three production tasks that run sequentially as part of the Detail Extractor job: IVNScanner, StatementsToIR, and DXLoader.
- Publish the Detail Extractor view files, which include a DDF file with the rules for extracting the table data, database table schema XML, and a statement XSLT style sheet to organize the extracted data the database table. You can use the editor of your choice to create the XML and XSLT files.

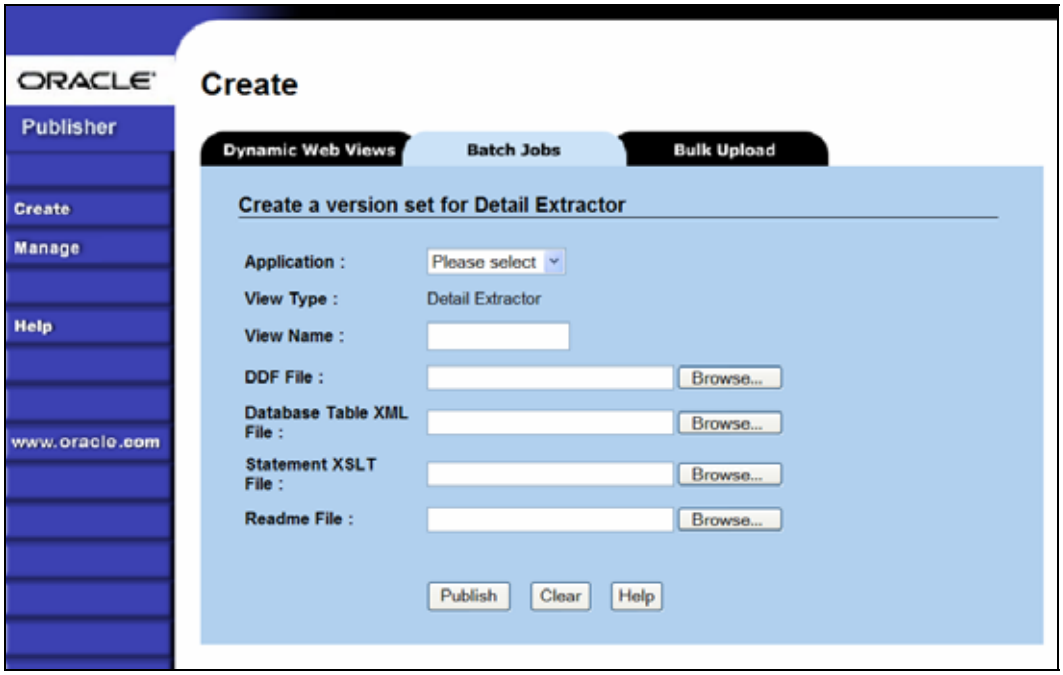
You can use the same DDF file for multiple Detail Extractor jobs; define all tables and groups you plan to upload to a database in the DDF.

Review the task and field configuration settings in this section to determine which options to set for the Detail Extractor job.

Tip: You can publish the required Detail Extractor view files (DDF, XML, and XSLT) at the time you create and configure the Detail Extractor job or you can publish these and all other required application files in bulk. Bulk publishing makes it easier to move application files between development, testing, and production servers. See Command Center Help for details.

To create and configure a Detail Extractor job:

1. On the Main Console, click the application name, listed under Applications in the table. The Edit Application screen appears.
2. Click **`Add New Job`**. eStatement Manager displays the Create New Job screen.
3. Enter a meaningful name for the Detail Extractor job. The job name must start with an alpha character. The rest of the name can be alphanumeric and can have underscores but no spaces. (Using the name of the table defined in the DDF for the job name can help you remember which data the job uploads.)
4. Select job type **`Detail Extractor`**.
5. If you plan to publish this application's files in bulk, skip to Step 17. Otherwise click **`Launch Publisher`**. See Command Center Help for information about bulk publishing.
6. Click **`Create`** on the Publisher Menu.
7. Click the **`Batch Jobs`** tab.
8. For the job type Detail Extractor, click **`0`** (Number of Auxiliary files). Publisher displays the Create a Version Set for Detail Extractor screen:



9 Select the application from the drop-down list.

10 Specify a view name for the job that uniquely identifies the view of table data the job extracts. (Using the name of the table defined in the DDF for the view name can help you remember which view maps to which table.) Do not reuse the name of a view given to another Detail Extractor job.

11 Select the DDF file for the Detail Extractor job.

12 Select the database table XML file you created for the view.

13 Select the statement XSLT style sheet.

14 Browse and select the readme.txt for this version set (if you've created one). See Command Center Help for details about the readme.txt.

15 Click **Publish**. If successful, Publisher displays the message "This Version Set has been Published Successfully" and displays details about the view files.

16 Close Publisher.

17 At the Create New Job screen in Command Center, click **Configure Job and Continue**. Command Center displays the Detail Extractor job configuration screen.

18 Specify the configuration parameters for each of the three tasks that run as part of the Detail Extractor job. Carefully read the descriptions of each task and field (below) to choose the values appropriate for your application and job.

19 When finished entering configuration parameters, click **Submit Changes and Schedule**. EStatement Manager submits the job configuration parameters and displays the Schedule screen. You can specify the Detail Extractor job schedule later. (You must schedule Detail Extractor to run after the Indexer job; Detail Extractor only uploads data for statements that have been indexed).

20 Click **Main Console**.

Task 1: IVNScanner

The IVNScanner task determines whether index data has been verified before letting the Detail Extractor job continue processing. This task is primarily intended for applications with a customized verification/audit application.

IVNScanner looks for a date processed in the Date Accepted (or Date Rejected) column in the volumes table. A volume is a data file that has been successfully processed by the Indexer job and referenced in the volumes table. The Indexer job must run before Detail Extractor. The Detail Extractor job processes one indexed volume at a time until IVNScanner finds no more newly indexed volumes listed in the volumes table.

If you selected "Intercept to Verify" in the Action on Index Volume option when configuring the AutoIndexVolAccept task in the Indexer job, you must use your customized verification/audit application to either accept or reject the indexed data before the Detail Extractor job can upload data from the indexed data input file. (You can optionally choose to run the job when the volume is rejected; this is useful in a test environment only.)

IVNScanner Task Input:	**IVNScanner Task Output:**
• A date processed value in the Date Accepted or the Date Rejected columns in the volumes table	• (None)

IVNScanner Task Configuration		
Field	**What to enter/select**	
Index Volume Status	Specifies that the job can proceed when a date appears in either the Date Accepted or Date Rejected column in the volumes table.	
	Accepted	(Default) Choose this option if you do not have custom verification/audit application or if you have one and you want to run the job only after a volume has been approved. If you choose this option, IVNScanner looks for a date in the Date Accepted column; if it contains a date, the Detail Extractor job proceeds.
	Rejected	Use this setting to continue the job if a volume has been rejected in the custom verification application (use this option for testing purposes only). If you choose this option, IVNScanner looks for a date in the Date Rejected column; if it finds a date, the job proceeds.
Scan Starting From (Number of Days)	Specify how many previous days' volumes to scan for; IVNScanner selects any volumes indexed on or between the current date and the number of days ago you specify.	

Task 2: StatementsToIR

The StatementsToIR task uses the DDF, database table schema XML, and the XSLT style sheet view files published with the job to extract the recurring detail content from the data input file and place the data into an Intermediate Representation (IR) file.

StatementsToIR Task Input:	StatementsToIR Task Output:
• Data source file in the output file path • Most recent DDF published in AppProfiles directory (not configurable) • The database table schema XML and the XSLT style sheet files published with the view	• An .IR file containing the extracted data, in the data file path

StatementsToIR Task Configuration	
Field	**What to enter/select**
View Name	The name of the view you published for this job. The Detail Extractor view consists of a DDF, a database table schema XML, and an XSLT style sheet.
Enroll Model	Specify the name of the enrollment resolver your application uses.
Output File Path	Specify the name of the application data directory where you want to create the IR file. The default is your data output directory (which you specify in the configuration of your Indexer job's Scanner task).

Task 3: DXLoader

The DXLoader task creates a new detail database table and uploads the detail data in the .IR file to the table using the database platform's SQL loader.

DXLoader Task Input:	DXLoader Task Output:
• .IR file	• A database table with content • .LOG, .CTL, and .CMD files (creates a .bad file if the SQLLDR or BCP fail)

DXLoader Task Configuration	
Field	**What to enter/select**
Load Method	Choose one of the following database load methods:

DXLoader Task Configuration		
Field	**What to enter/select**	
	Direct load	Stores data directly to the database. This option locks the Detail table, loads the information, and ends the call to the database. No sharing of tables is allowed during this process. A direct load is usually the fastest method. Oracle recommends this method for Oracle and SQL Server.
	Conventional load	Uses Insert statements, one row at a time. Performs multiple selects and inserts on the table at once, but does not lock up the database and lock out Web users. A conventional load is usually slower than a direct load. Oracle recommends this method for DB2.

Publishing Your Application's Dynamic Web Views

After creating and configuring your application jobs, you must publish each dynamic Web view in your application.

You can publish Web views (version sets) individually with the rest of an application's files in bulk.

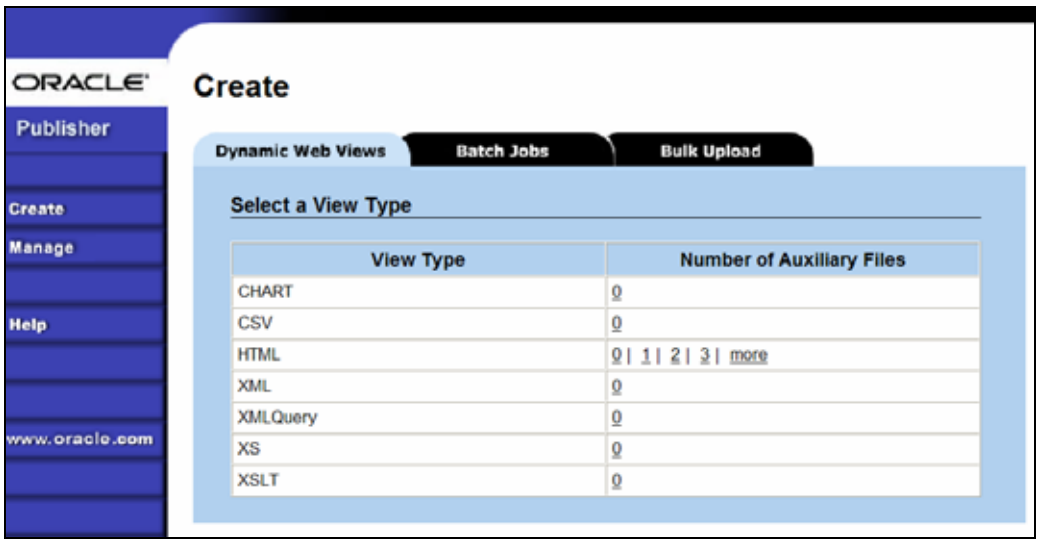
"Publishing" a dynamic Web view identifies each file belonging to a particular view, and lets you give the view a name. When a user clicks a link to their statement ("running" a dynamic Web view), the link identifies the set of design files to use to extract and present the user's statement data.

Publishing dates the set of view files with a timestamp. Publishing a view is also called "creating a version set." A version set is a dated set of design files.

Tip: You can publish dynamic Web views individually or in bulk. Bulk publishing makes it easier to move application files between development, testing, and production servers. You can bulk publish dynamic Web views along with all other required application files when setting up a new application, or bulk publish multiple version sets as needed to update an active application. See Command Center Help for details.

To publish a single dynamic Web view for an application:

1. Click **`Publisher`** on the Command Center Main Console. eStatement Manager launches Publisher.
2. Click **`Create`** on the Publisher menu.



3 To publish a dynamic HTML version set, under "Number of Auxiliary Files" click the number of auxiliary HTML templates the view uses (in addition to one); if the view uses one HTML file, click 0. If the view uses two HTML template files, then click 1, etc.

To publish a CSV, XML, Chart, XSLT, or XML Query view, click the **0** under "Number of Auxiliary Files" for the job type. (To create an XS view, also click the number of auxiliary files in the view.) Publisher displays the Create a Version Set screen for the view type you're publishing:

4 Select the application name from the drop-down list.

5 Enter the view name; this is the specific name the Web browser looks for in the code. This view name is customized in your JSP and HTML pages; consult your design team for details.

6 Select the design files to publish. You publish the following files for each type of dynamic Web view:

HTML	A DDF, ALF, and one or more HTML files
XS	One or more XSL files
CSV	A DDF and a TOK file
XML	A DDF file
Chart	A Properties file
XSLT	An XSLT file and an application DDF
XML Query	An XML file

7 Browse and select the readme.txt for this version set (if you've created one). See Command Center Help for details about the readme.txt.

8 Click **Publish**. If successful, Publisher displays the message "This Version Set has been Published Successfully" and displays details about the view files.

9 If you have any additional dynamic Web views for your application, click **Create** and continue to publish version sets for those views.

Once your application and jobs are defined and you've published your dynamic Web views, you can proceed to schedule jobs for live production.

4 Publishing and Using Version Sets

When to Publish New Version Sets

As part of the initial process of setting up your eStatement application, you published the original design files that present an application's statements and generate email notifications.

A version set is a dated set of design files which eStatement Manager uses to present a user with an online view of a statement. "Publishing" a version set using the eStatement Manager Publisher tool identifies each design file belonging to a particular view and moves the files from the design environment to your application server. You give the view a name, and Publisher timestamps the version set for further identification.

Your organization might occasionally find it necessary to modify application design files to apply new business logic, text messages, logos, advertisements, other elements of an online statement, or to accommodate a new data input file format. To implement the new design files in your production environment, you must publish the files, creating new version sets.

It is only necessary to publish design files again for an application if you update the files for any reason. You cannot update or overwrite an existing version set; you must create and publish a new one.

If you create a new version set to accommodate changes in the application data input file, you must also publish a new Indexer version set (DDF).

When you publish a new version set for the view, eStatement Manager proceeds to use the new version set with Indexer jobs until you publish another one. Publishing a new version set on the same day you run the Indexer job makes it easier to keep track of when design changes take effect.

It is possible to publish multiple version sets on the same day an Indexer job is run, or on subsequent days.

Tip: You can publish version sets for one or more applications in bulk. Bulk publishing makes it easier to move multiple new application files between development, testing, and production servers. See Command Center Help for details.

Each time you publish a new version set, eStatement Manager:

- Creates a timestamp version directory called *AppName/ViewType/ViewName/Timestamp*
- Adds each file in the version set to the directory
- Creates a reference in the database for the version set

For details on what files you must publish for each type of job, see "What Files Do I Need to Publish?" on Page 19.

You can publish new version sets individually or in bulk. When you upload a bulk file, Publisher adds to any existing views and view types already in an application. See Command Center Help for details on bulk publishing.

Note that you do not need to publish files for Purge App and Purge Logs jobs.

CAUTION: Do NOT attempt to edit the XML DDF directly, or the resulting DDF may be unusable. If you receive the error message "Error in reading XML DDF, please see the Error Log File!" when publishing a DDF, the internal validation failed and the DDF may have been altered.

To publish a new dynamic version set:

See "Publishing Your Application's Dynamic Web Views" on Page 52 for instructions.

Viewing Job Output

You can view application output online to verify that your application and system are working properly.

You must first enroll through the eStatement Manager Enrollment Module; enrolling identifies each user to an online billing service provider. After you've enrolled successfully, you can log into the Enrollment Module and view a statement.

To enroll for online statement viewing:

1. Run the Indexer job and publish the design files you want to view in live retrieval.
2. After the Indexer job executes successfully, connect to the eStatement Manager Enrollment Module; enter the following URL in your browser, substituting your server and application names:
   `http://YourMachineName:PortNumber/Sample/User?app=UserMain&jsp=/user/jsp/HistoryList.jsp&ddn=ApplicationName`

   (For testing purposes, you can view the Sample application.) eStatement Manager displays an enrollment login page.
3. Click the **Enroll Now** link. An empty subscription page appears.
4. Enter the customer number and other enrollment data. (If you are viewing the National Wireless sample application, you can use sample customer account numbers 0331734, 4191463, or 8611250.)
5. Click **Submit** to save the subscription information or **Reset** to clear the text fields.
6. Return to the Enrollment login page and enter your username (Subscriber ID) and password. This is the same username and password combination you entered during enrollment.
7. Click **Submit**. A sample statement summary page appears.
8. To view details, click **View**.
9. Return to the statement summary page and click **Logout**.

5 Using Job Alerts

Overview

The job alerts feature lets you send an email notification when a job successfully completes processing, fails processing, or both.

To implement email notification alerts for a job, you must:

- **Create one or more alert groups**. Alert groups let you define a list of email addresses (contacts) to receive email notification alerts. You can create multiple lists to use with different jobs, or just one alert group for many jobs.
- **For each alert group, create and configure one or more alert profiles.** Alert profiles let you define a list of applications and jobs. You then associate alert groups with alert profiles to specify which contacts to notify for each job.

You use the Scheduler to disable/enable alerts for a particular job, and if necessary, to override the Job Alert Profiles set up for the job and specify an alert group or manually enter a list of contacts instead.

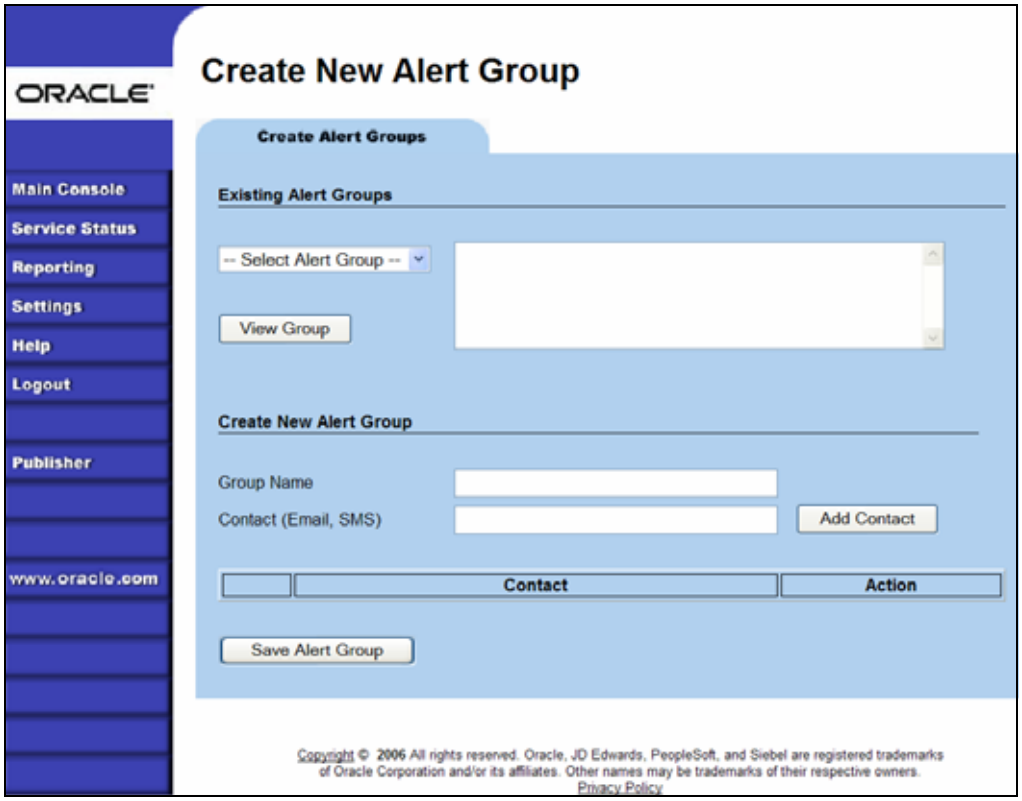
Creating, Editing, and Deleting Alert Groups

Creating Alert Groups

To create a new alert group you must provide the group a name and add one or more contact email addresses.

To create a new alert group:

1. Click **Settings** from the Command Center Main Console. The Settings screen appears.
2. Click the **Alert Settings** tab.
3. Click the **Create Alert Groups** tab. The Create New Alert Group screen appears.



4. In the Group Name field, type a name for the new alert group.
5. Add the contact email address where you want the system to send the alert email message.
6. Click **Add Contact**. The system adds the contact to the list. (Note that you can edit or delete a contact in this table using the **Edit** and **Delete** options in this table.)
7. Continue adding contacts to the new alert group as needed.
8. Click **Save Alert Group**.
9. You can now display the list of group contacts. Under Existing Alert Groups, select the group from the drop-down list and click **View Group**.

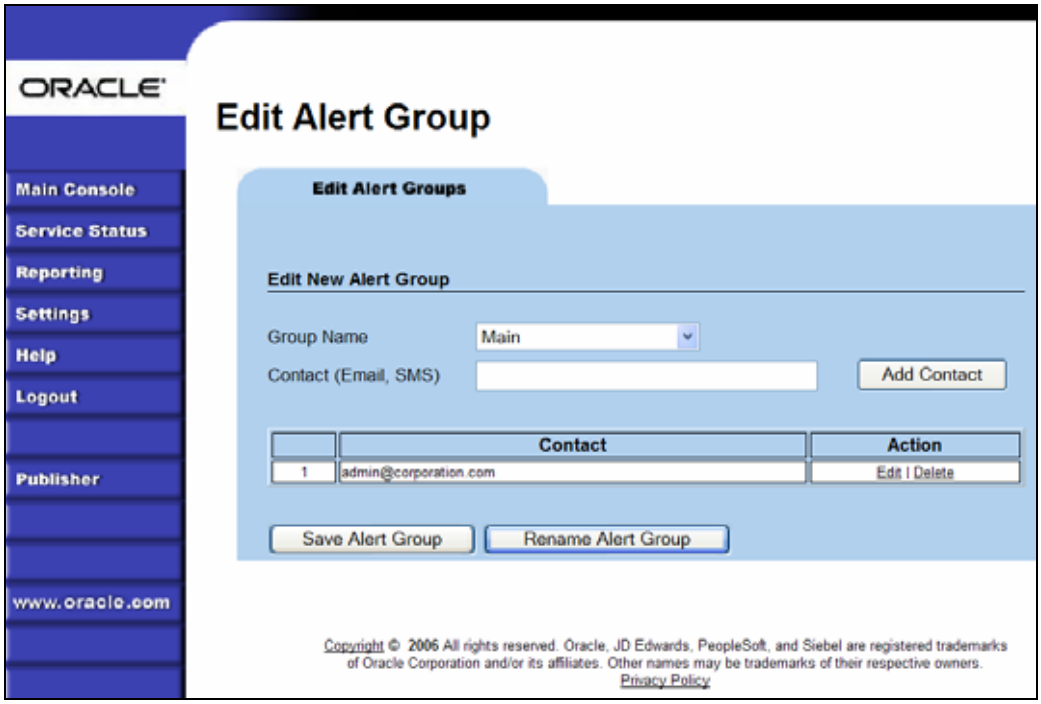
Editing Alert Groups

Editing alert groups lets you rename a group, or view, add, edit or delete contacts from a group.

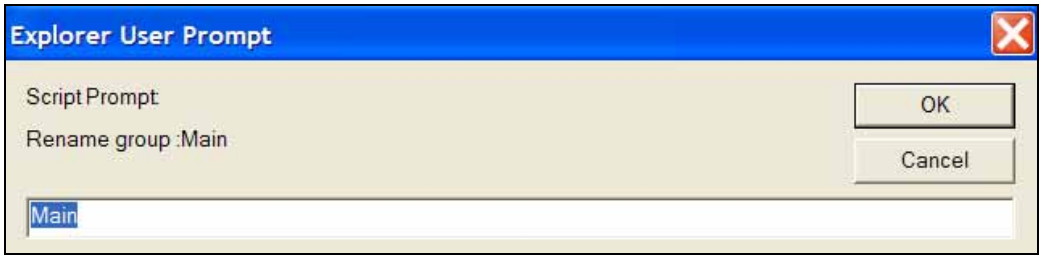
To edit an existing alert group:

1. Click **Settings** from the Command Center Main Console. The Settings screen appears.
2. Click the **Alert Settings** tab.
3. Click the **Edit Alert Groups** tab. The Edit Alert Group screen appears.

4 Choose an alert group to edit from the drop-down list. A list of current contacts for the selected group appears.



5 To change the group name, click **Rename Alert Group**, type a new name, and click OK.



To add a contact to the group, type the email address in the Contact field and click Add Contact.

Click **Edit** or **Delete** to edit or delete a contact.

6 Select **Save Alert Group** option to save your changes.

Deleting Alert Groups

You can delete alert groups from the system.

To delete an alert group:

1 Click **Settings** from the Command Center Main Console. The Settings screen appears.

2 Click the **Alert Settings** tab.

3 Click the **Delete Alert Groups** tab. The Delete Alert Group screen appears.

4 Select the group you want to delete from the drop-down menu.

5 Click **Delete Alert Group**. At the confirmation prompt verify the group and click **OK**.

Creating Job Alert Profiles

A job alert profile lets you create a list of applications and jobs to associate with an alert group. One alert group can have multiple alert profiles associated.

You can choose to base a profile on:

- **All Jobs** – Includes all jobs for all applications (DDNs) in the profile.
- **Selected Job Types** – Lets you select a list of specific job types (for all DDNs).
- **Selected DDN's** – Lets you select a list of DDNs (for all job types).
- **Selected Job Types in a Particular DDN** – Let you choose particular job types for a specific DDN.

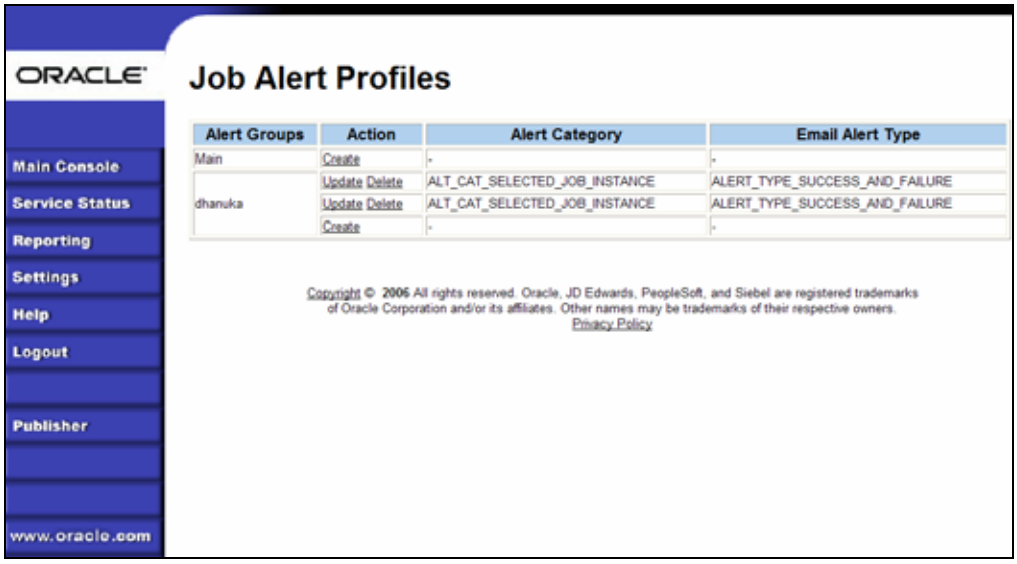
For each profile you must specify alert types. The alert type indicates when to send email notification to the associated alert group contacts:

- **Success** – When the job completes successfully.
- **Failure** – If the job fails.

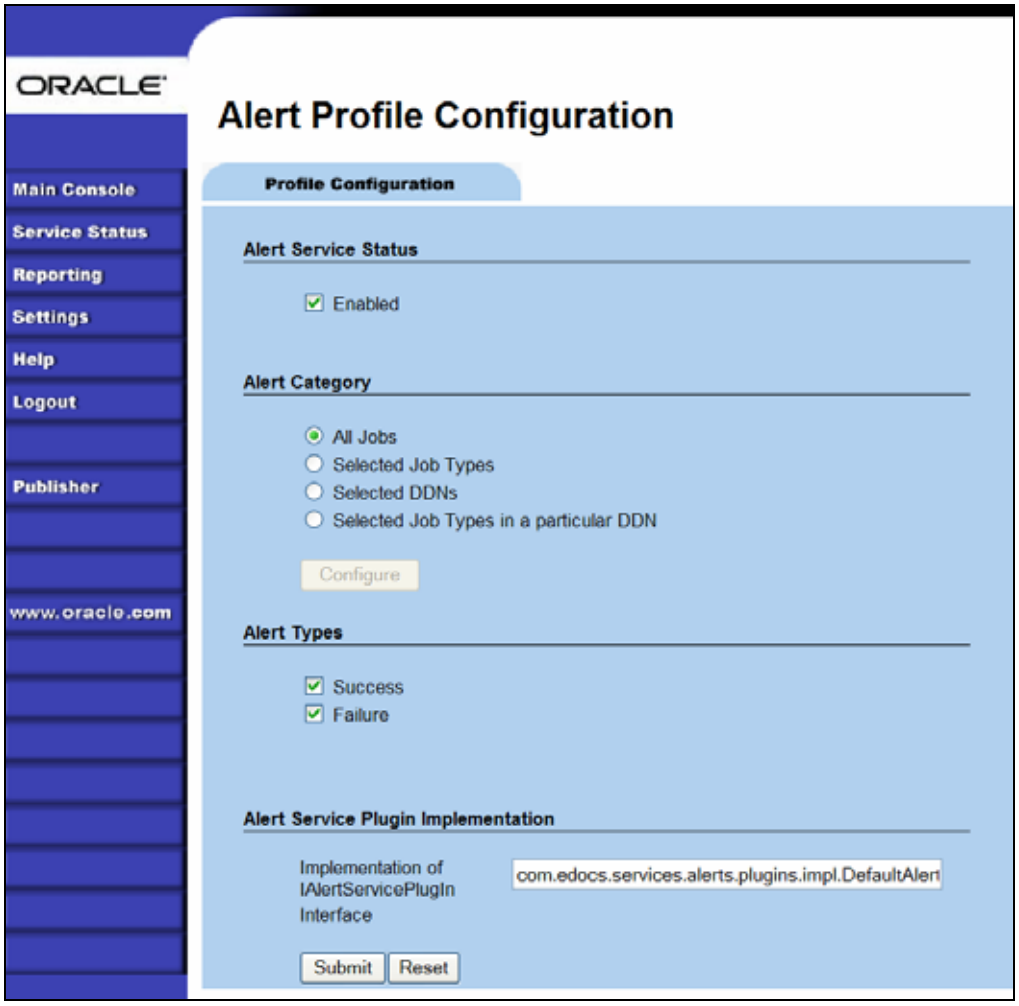
You can choose both Success and Failure alert types for a profile.

To create a new job alert profile for an alert group:

1. Click **Settings** from the Command Center Main Console. The Settings screen appears.
2. Click the **Alert Settings** tab.
3. Click the **Configure Alert Profiles** tab. The Job Alert Profiles screen appears.



4. Under the Action column for the alert group, click **Create**. The Alert Profile Configuration screen appears.

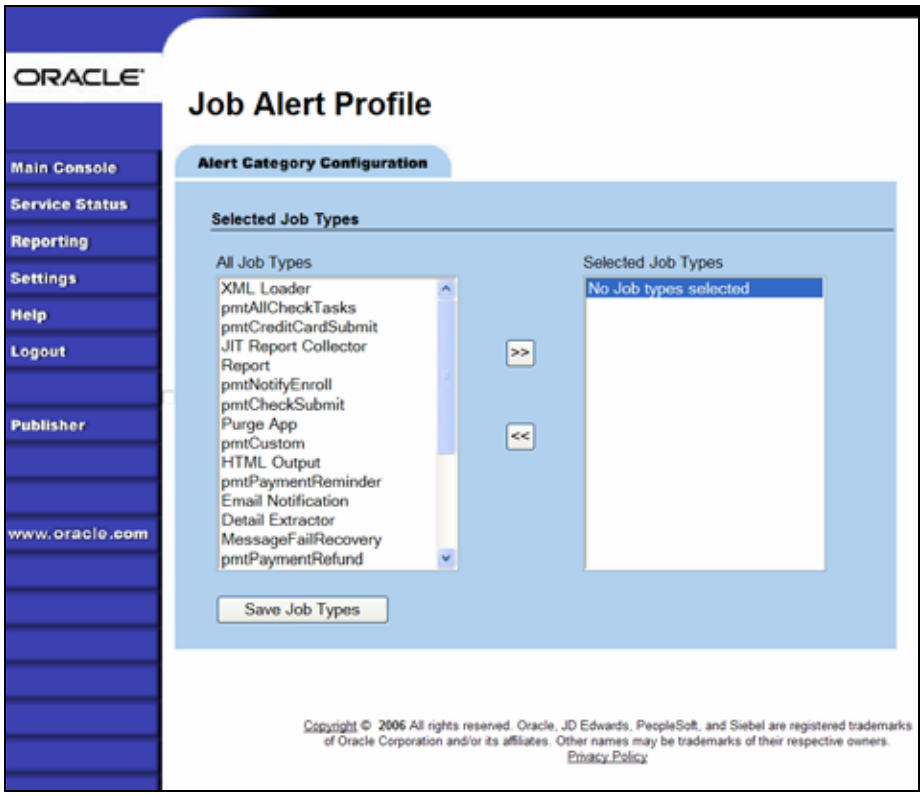


5 Make sure the Alert Service Status is enabled (default).

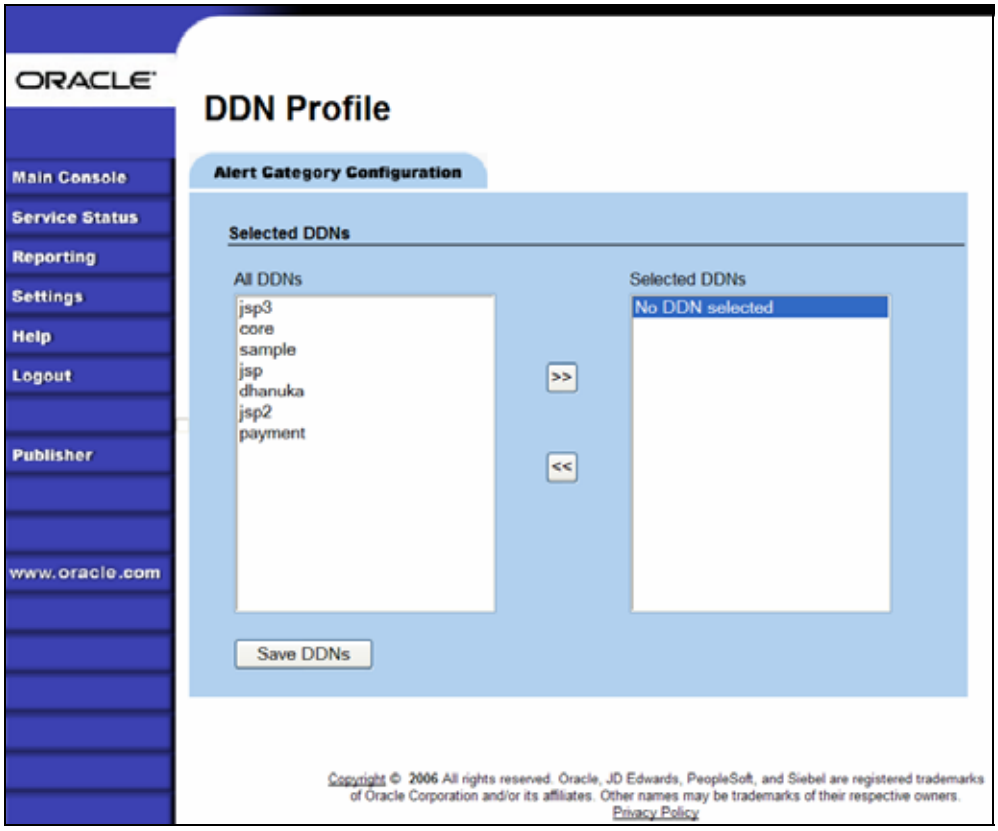
6 Under Alert Category, choose one of the following options and follow any additional steps noted. (Use the **Reset** button if you need to clear the screen selections at any point).

All Jobs - To send alerts for all jobs and all DDNs (applications). (This option is selected by default).

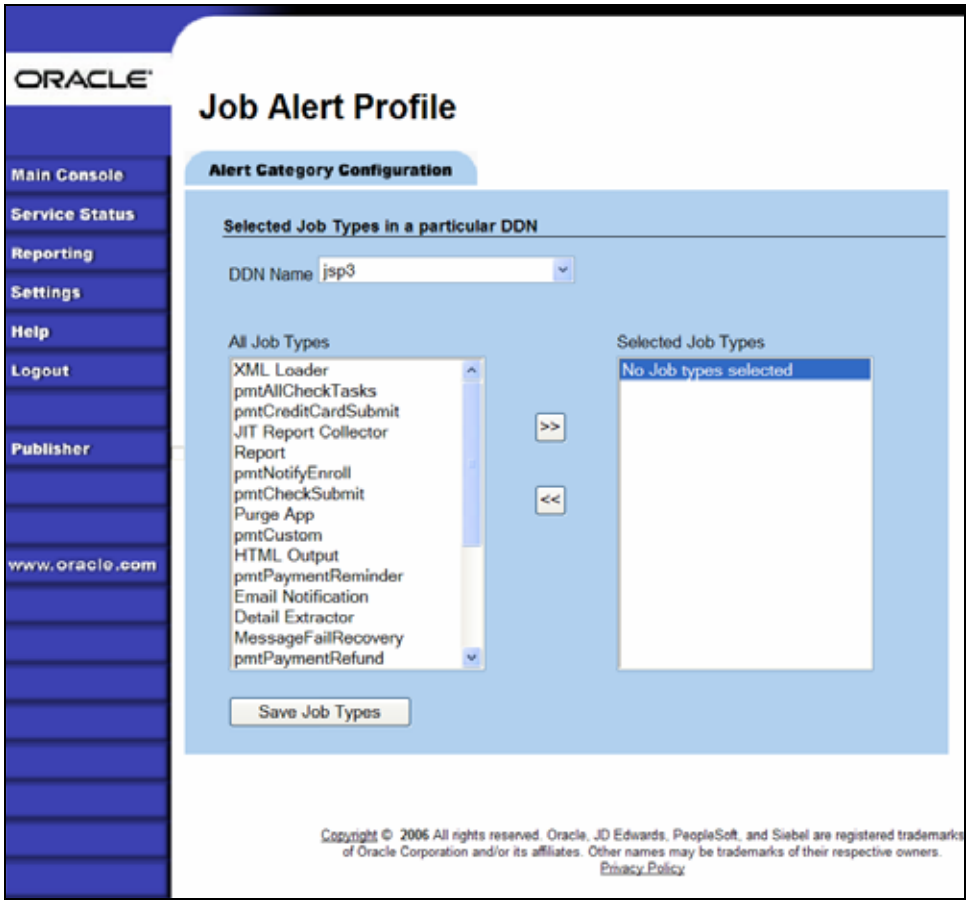
Selected Job Types - To select specific job types (for all DDNs) to send alerts. Select this option then click Configure. The Alert Category Configuration screen appears showing all job types in the Command Center. To select a job type, highlight it and click the double right-arrow button. (To deselect any job types, highlight the job type under "Selected Job Types" and click the double left-arrow button.) Click **Save Job Types**.



Selected DDNs - To select specific DDNs (for all job types) to send alerts. Select this option then click Configure. The DDN Profile screen appears showing all DDNs defined in the Command Center. To select a DDN, highlight the name and click the double right-arrow button. (To deselect a DDN, highlight the name under "Selected DDNs" and click the double left-arrow button.) Click Save DDNs.



Selected Job Types in a Particular DDN - To select job types in a specific DDN to send alerts. Select this option then click Configure. The Alert Category Configuration screen appears. Choose a DDN from the drop-down list. To select a job type, highlight the name under "All Job Types" and click the double right-arrow button. (To deselect a job type, highlight the name under "Selected Job Types" and click the double left-arrow button.) Click Save Job Types.



7 Choose one or both alert types for the alert group profile: Success (when the job completes successfully) or Failure (if the job fails).

8 Specify the name of the custom alert service plugin in the **Implementation of IAlertServicePlugin Interface** field to use for this profile (if one exists).

9 Click **Submit** to save the alert group profile configuration.

Updating a Job Alert Profile

To update a job alert profile:

1 Click **Settings** from the Command Center Main Console. The Settings screen appears.

2 Click the **Alert Settings** tab.

3 Click the **Configure Alert Profiles** tab. The Job Alert Profiles screen appears.

4 In the Action column for the job alert, click **Update**. The Alert Profile Configuration page appears.

5 Edit the alert profile as needed and click **Submit**.

Deleting a Job Alert Profile

To delete a job alert profile:

1. Click **Settings** from the Command Center Main Console. The Settings screen appears.
2. Click the **Alert Settings** tab.
3. Click the **Configure Alert Profiles** tab. The Job Alert Profiles screen appears.
4. In the Action column for the job alert, click **Delete**. The system removes the job alert profile.

Configuring Job Alerts in the Scheduler

By default, the scheduler is automatically set up to enable alerts for all jobs and uses the configurations you define in the Job Alert Profiles screen.

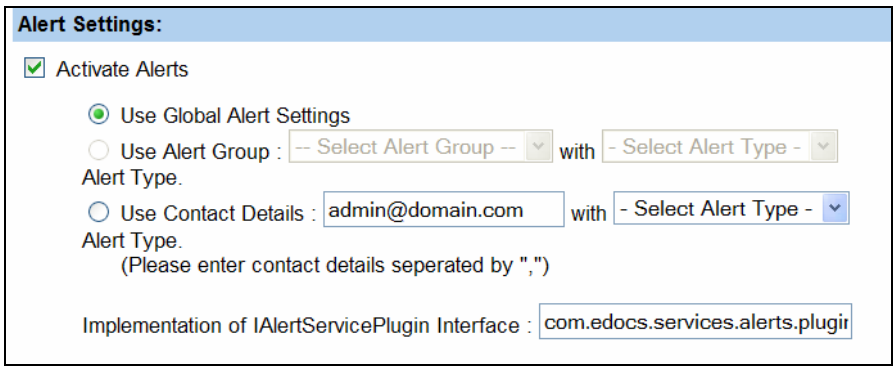
You can override these "global" configurations for a particular job at any time by changing the alert settings in the Scheduler.

For a particular job you can:

- Turn off the alert service.
- Override the global Job Alert Profile configuration information and select a particular alert group and alert type to use instead.
- Override the global Job Alert Profile configuration information and manually enter a list of email contacts and alert type to use instead.
- Specify the name of a customized interface plug-in to use for the job, if you have created one.

To change alert settings for a job in the Scheduler:

1. On the Command Center Main Console, click the status under the Next Run column for the particular application job. The Schedule Job screen appears.
2. Under the Alert Setting section, make sure **Activate Alerts** is selected.



3 Choose one of the following alert options to use for the job:

Use Global Alert Settings (Default) – Uses the alert profile configurations and associations with alert groups defined on the Job Alert Profiles screen.

Use Alert Group - Select an alert group and alert type (Success & Failure, Success Only, or Failure Only) to use for this job, overriding the profile configurations and associations with alert groups defined on the Job Alert Profiles screen.

Use Contact Details - Enter one or more specific email addresses and an alert type to use for this job, overriding the profile configurations and associations with one or more alert groups defined on the Job Alert Profiles screen.

4 If you have a customized IAlertServicePlugin and want to override one defined in the Job Alerts Profile configuration for this job, enter that file name in the **Implementation of IAlertServicePlugin Interface** field.

5 Click **Save Schedule**. The next time the job runs it will use these configurations.

6 Blackout Date Job Scheduling

The blackout date job scheduling feature lets you define days during which the system ceases job processing.

To implement blackout date job scheduling, you create a calendar and specify which days are blackout dates. You can create multiple calendars, though only one calendar can be in effect at a time.

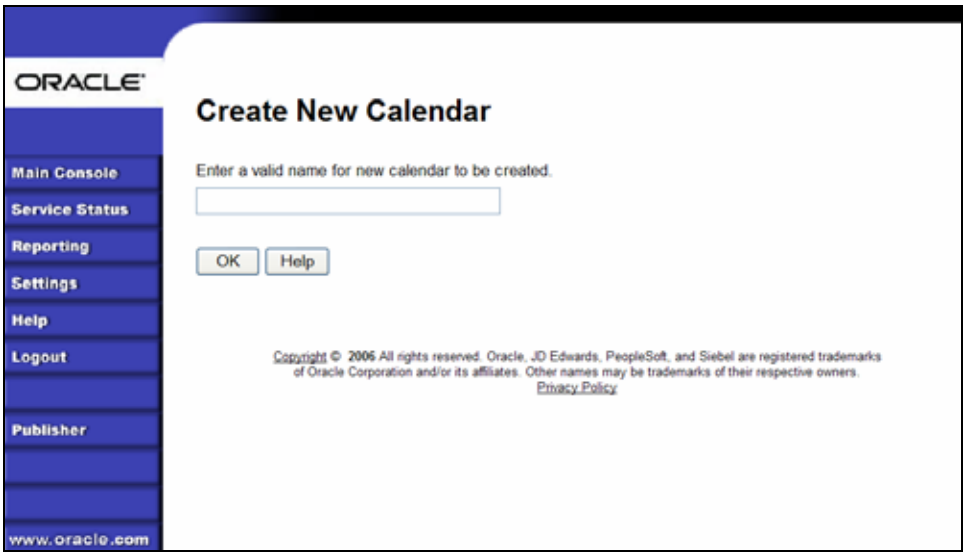
For each Command Center job you configure, you associate the job's schedule with a particular calendar using Scheduler. Using a blackout date calendar with jobs is optional.

You can copy the data from an existing calendar to start a new one, and edit and delete calendars as needed.

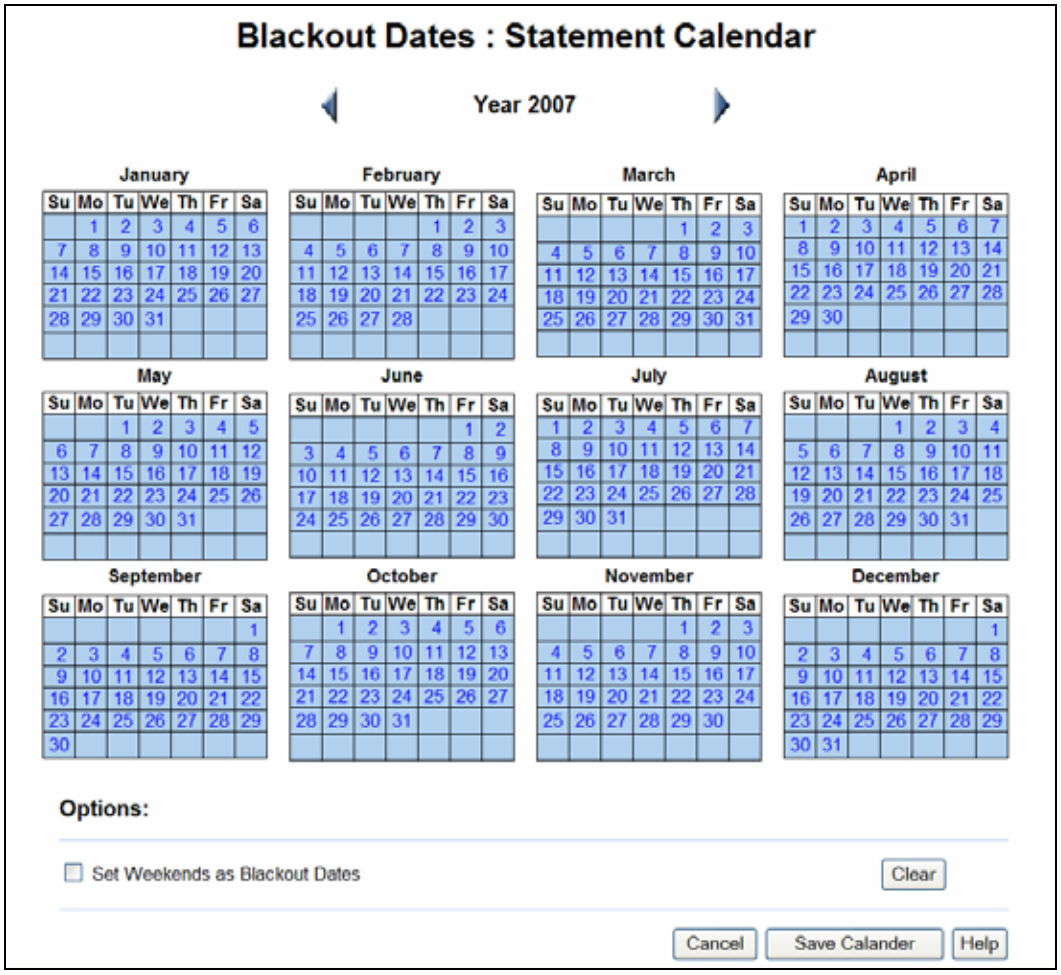
Creating a New Blackout Date Job Calendar

To create a new blackout date job calendar:

1. Click **Settings** from the Command Center Main Console. The Settings screen appears.
2. Click the **Calendar Settings** tab.
3. Click the **Create New Calendar** tab. The Create New Calendar screen appears.



4. Type a name for the calendar and click **OK**. The Calendar Page appears showing a calendar for the current year.



5 Click a date to toggle it on/off as a blackout date.

6 You can select **Set Weekends as Blackout Dates** to automatically select all weekend days for blackout job processing.

7 Use the **Clear** button if you want to clear all your selections.

8 Click the right and left arrow buttons to display the previous or next year's calendar.

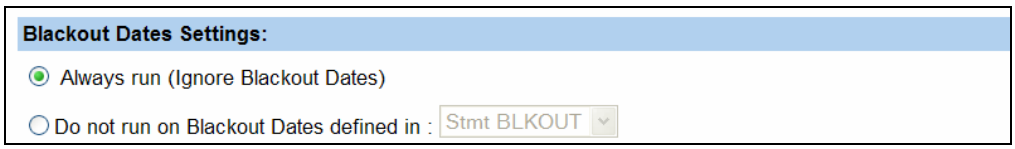
9 Click **Save Calendar** to save your changes.

Applying a Calendar to a Job Schedule

To apply a blackout date calendar to a job schedule:

1 On the Command Center Main Console, click the status under the Next Run column for the particular application job. The Schedule Job screen appears.

2 Select the **Do not run on Blackout Dates defined in** option and choose a calendar from the drop-down list.



3 Click **Save Schedule**. The system now ceases processing the selected job on the blackout dates in the selected calendar.

Copying a Calendar

You can create multiple calendars, basing a new calendar on the data in an existing one.

To copy the blackout dates from an existing calendar:

1 Click **Settings** from the Command Center Main Console.

2 Click the **Calendar Settings** tab.

3 Click the **Copy Calendar** tab. The Copy Calendar screen appears.

4 Choose the calendar with the settings you want to copy from the drop-down list.

5 Type a name for the new calendar and click **OK**. The Calendar Page appears showing the current year's data in the new calendar.

6 Edit the calendar, if needed, and click **OK**. The system saves the new calendar.

Editing a Calendar

You can change the selected blackout dates on an existing calendar.

To edit a saved calendar:

1. Click **Settings** from the Command Center Main Console. The Settings screen appears.
2. Click the **Edit Calendar** tab.
3. Choose a calendar to edit from the drop-down list and click **OK**. The selected calendar appears showing the current year.
4. Edit the calendar as needed and click **Save Calendar**.

Deleting a Calendar

You can delete blackout date job calendars from the system. Be sure to unassociate a calendar from any jobs using the Schedule Job screen. You cannot delete a calendar that is actively associated with a job schedule.

To delete a blackout date job calendar:

1. Click **Settings** from the Command Center Main Console. The Settings screen appears.
2. Click the **Delete Calendar** tab. The Delete Calendars screen appears.
3. Click the **Confirm Delete** check box to any calendars you want to delete. Review your selections carefully. You can delete a calendar only if the screen indicates there are "No Associated Jobs" for the calendar.
4. Click **Delete Selected Calendars**. The system removes the selected calendars.

7 Appendix A: Glossary

eStatement Manager Terms and Glossary

This glossary defines eStatement Manager terms as well as technology and general business terms related to electronic presentment and payment.

Account Number	The identification of an individual document in a source file. The account number indicates the identity of the individual or business associated with the account information.
ACH	Automated Clearing House. A network for Electronic Funds Transfer, governed by the National ACH Association (NACHA) and the U.S Federal Reserve Bank. The ACH Network is used for all kinds of EFT transactions, including direct deposit of paychecks and monthly debits for routine payments to vendors. The ACH network is separate and distinct from the various bankcard networks that process credit card transactions. ACH transactions are conducted among participating financial institutions by the transfer of ACH files in batch mode, which can take up to 72 hours before the money is actually transmitted. In Payment, an ACH return file is sent if there are insufficient funds in the account or if there are other errors or problems with the transaction. (Term used with Payment.)
Addenda Record	ACH record type data needed to completely identify an account holder or to provide information concerning a transaction that carries supplemental payment. (Term used with Payment.)
AIX	Advanced Interactive Executive. IBM's version of UNIX, which runs on 386 and higher PCs, RS/6000 workstations and 390 mainframes. It is based on AT&T's UNIX System V with Berkeley extensions.

ALF	Application Logic File. An ALF contains the rules to present the data extracted from the original data source in a customized HTML format. An ALF is created during application design and development using the eStatement Manager Composer tool. The ALF (which is in XML format) is then used along with the DDF during live statement retrieval to display extracted account data on the Web, in email, etc. The ALF also contains business logic (conditional statements that consider current statement data) for marketing and other customization purposes. In addition to dynamic statement presentment, ALFs are also developed and used for the composition of static output, most commonly notification emails. An application can have multiple ALFs for use in presenting different statement views. See also **View**.
Anchor	An additional (and optional) condition placed on a DDF extraction rule to ensure the proper data is extracted or located. Anchors can be defined on any DDF extraction rule, including fields, markers and page styles. If an anchor is defined, the system looks for the anchor pattern first and then the defined element. Anchors are useful when the data to be located is too generic to be detected by a pattern or it is found within a large range of rows and columns. See also **Table Anchor**.
API	Application program interface. A set of routines, protocols, and tools for building software applications. A good API makes it easier to develop a program by providing all the building blocks. A programmer puts the blocks together.
Application	A customized set of eStatement files (DDFs, ALFs, Statement HTML templates, Group HTML templates and images) created exclusively to extract and present online statements for a service provider. A custom application must be designed for each input data source. Each application is given a unique name. A single service provider can have more than one application if they present data from input data sources with different formats or content.
Application Name	The name of a particular application. eStatement expects a DDF and ALF named after the application when dynamically composing the first statement view. The main application directory must also be named after the application. See also **DDN**.
ASP	Active server page. A Web server technology from Microsoft that allows for the creation of dynamic, interactive sessions with the user. An ASP is a Web page containing HTML and embedded programming code written in VBScript or Jscript. It was introduced with Version 3.0 of Microsoft's Internet Information Server (IIS). When an ASP page is requested by the browser, IIS executes the embedded program. ASPs are Microsoft's alternative to CGI scripts and JavaServer Pages (JSPs), which allow Web pages to interact with databases and other programs. Third party products add ASP capability to non-Microsoft Web servers. ASP can also refer to Application Service Provider. See **Service Provider**.
Authentication	The process by which a Web application (such as an eStatement application) verifies the identity of an individual Web user. eStatement applications must verify the identity of an enrolled user before they can view account information.

Business Logic	In eStatement Manager, the conditional statements added to an ALF during application design and development. Conditional statements are written to create customized statements based on the particular financial and personal activity of each statement recipient. Business logic lets you display alternative messages or implement targeted advertising campaigns under specified conditions. Also refers to the Java code that implements the functionality of a business application. In the EJB model, this logic is implemented by the methods of an enterprise bean.
CCD	Cash Concentration and Disbursement. An automated corporate payment used primarily for the intra-company concentration or disbursement of funds. Payment supports this type of ACH payment. (Term used with Payment.)
Column	A unique data field that appears within a table. The columns of a table are usually related to each other in some logical way. Related table column occurrences usually appear on the same row, although this is not required. For example, in a table of telephone call detail, there could be columns for "Number Dialed," "Call Duration," and "Cost of Call." (In previous releases of eStatement Manager, columns have been referred to as record fields.)
Composer	The Windows-based, GUI tool used to create the rules for mapping data extracted from an input file to an HTML template for display on the Web, email, etc. The Composer displays the HTML template in WYSIWYG format and lets designers drag and drop placeholders for data elements into the template. The Composer also lets designers define business rules (build conditions) for marketing purposes. The main output of the Composer is an Application Logic File (ALF). The secondary output of the Composer is HTML templates, edited to include placeholder tags for the placement of extracted data elements and the execution of business logic statements. Composer is used along with DefTool to design and develop an eStatement application.
Container	A Java entity that provides life cycle management, security, deployment, and runtime services to components. Each type of container (EJB, Web, JSP, servlet, applet, and application client) also provides component-specific services.
CORBA	Common Object Request Broker Architecture. An architecture that enables pieces of programs, called objects, to communicate with one another independent of which programming language they were written in or what operating system they're running on.
CSR	Customer Service Representative. A CSR can be employed by the service provider or other entity.
CTE, CTX	Corporate Trade Exchange. An ACH Standard Entry Class code type used for corporate to corporate payments. CTX can transmit up to 9,999 addenda records in ANSI X12 syntax (EDI). (Term used with Payment.)

DDF	Data Definition File. The DDF contains the rules for finding and extracting data from an application's input data source. An application designer creates a DDF file using eStatement Manager's DefTool during the initial system design. It is created exclusively for use with a service provider's particular data source format and application. DDFs are used on a regular basis during statement live retrieval to extract account data for display on the Web. The DDF is also used by the Indexer task to prepare a data source and the entire eStatement application for live statement retrieval.
DDF Namespace	The collection of data elements defined for extraction in a DDF, including field, table, and group names. Most data elements are explicitly named, but some could be implicitly named (for example, the rows within a detail block might not have individual names). The contents of an application DDF Namespace appear in the left pane of the Composer screen, within the Definition tab. A designer can click and drag elements to the HTML view on the right.
DDN	Data Definition Name. The name of a particular application (internal to eStatement Manager). eStatement Manager expects a DDF and ALF named after the DDN when dynamically composing the first statement view. The main application directory must also be named after the DDN. See also **Application Name**.
DefTool	The Windows-based, GUI tool used to create the rules for finding and extracting data from a data source. DefTool's interface lets designers view the data source content in digitized format and graphically identify the elements to extract. The output of DefTool is a Data Definition File (DDF). DefTool is used along with the Composer tool to design and develop an eStatement application.
Detail Extractor	A type of eStatement Manager batch job used to extract and upload data from the data input file to an eStatement Manager database table. This data requires custom application functionality to extract the data from the database to merge with statements or use in any other way. For the Detail Extractor job you publish a DDF file, a database table schema XML file, and an XSLT style sheet.
Deployment	The process by which software is installed into an operational environment.
Distributed Application	An application made up of distinct components running in separate runtime environments, usually (in the eStatement Manager environment) on similar platforms connected via a network. Typical distributed applications are two-tier (client/server), three-tier (client/middleware/server), and n-tier (client/multiple middleware/multiple servers).
DOCID	An API-level string created by eStatement Manager to uniquely identify a particular document, or statement.
Document	A single statement or set of account data in an input data source. An input data source typically consists of many individual documents. Also used to describe a Web page composed by eStatement Manager to display a single user's statement or view of account data.

Document Style	A delimiter used to locate the first page of each statement in a data source. The Document Style data string is defined during application design to enable eStatement Manager to identify a new statement in the data source. If the Document Style is found on a page, it indicates with certainty that this is the first page of the next statement (delimiting statements in the data source). The Document Style definition consists of a name, pattern, and the row and column coordinates within which eStatement Manager must search for the start of the identifying data string.
Dynamic Paging	The ability to dynamically spread repeating data across multiple HTML pages including the ability to navigate the resulting pages. (Dynamic paging does not correspond to physical pages identified in the source data file.)
Dynamic Web View	A view used to create a dynamic statement presentation on the Web. Dynamic view types include HTML, CSV, XML, Chart, XSLT, and XML Query. You publish dynamic Web views.
eStatement Manager	Foundation eaSuite software product that provides secure online account management and presentment systems capable of using a variety of input data formats.
Payment	The eaSuite software product that provides electronic payment capabilities to an eStatement Manager system and lets the end-user manage and track payments. Payment supports a wide range of electronic payment methods including CheckFree, credit card and ACH transactions.
eaSuite	A set of Oracle Self-Service online presentment products, including eStatement Manager and add-on products.
EBPP	Electronic bill presentment and payment.
EJB	JavaSoft Enterprise JavaBeans. A specification defining a component architecture for building distributed, object-oriented business applications in Java. Enterprise JavaBeans encapsulate business logic. EJB developers can develop their own business components (such as invoices, bank accounts, and shipping routes), called enterprise beans, or purchase them from third-party vendors. Enterprise JavaBeans run in a special environment called an EJB container.
EJB Container	A container that implements the EJB component contract of the J2EE architecture. This contract specifies a runtime environment for enterprise beans that includes security, concurrency, life cycle management, transaction, deployment, and other services. An EJB container is provided by an EJB or J2EE server.
EJB Server	Software that provides services to an EJB container. For example, an EJB container typically relies on a transaction manager that is part of the EJB server to perform the two-phase commit across all the participating resource managers. The J2EE architecture assumes that an EJB container is hosted by an EJB server from the same vendor, so does not specify the contract between these two entities. An EJB server can host one or more EJB containers.
Email Notification	A type of eStatement Manager batch job used to dispatch secure email notifications to enrolled users after a statement has been processed. You must publish a view with each EmailNotification job.

Enrollment For Payment	The process by which an end-user becomes enrolled in a service provider's system for the purpose of paying bills. The enrollment process involves providing user-profile information that uniquely identifies the individual to an online billing service provider.
Enrollment For Presentation	The process by which a Web user becomes enrolled in a service provider's system for the purpose of viewing interactive or notification-based statement presentation. The enrollment process involves providing user-profile information that uniquely identifies the individual to an online billing service provider.
Enterprise Bean	A component that implements a business task or business entity; either an entity bean or a session bean. Also called EJB.
ERP	Enterprise Resource Planning. An integrated information system that serves all departments in an enterprise. ERP evolved from the manufacturing industry, and implies the use of packaged rather than proprietary software, although ERP modules can possibly be modified using a vendor's proprietary tools or a programming language (standard or proprietary).
Field	An independent data element within a statement that stands apart from any repeating line item detail. It is often used to isolate account information such as an account number, customer name, and summary financial information, such as total amount due.
FIDS	Formatted Input Data Stream. A type of data file created for an alternative method of generating a DDF automatically. A FIDS file contains extracted source data and related metadata. An eStatement Manager product extension uses the FIDS to automatically generate extraction rules (the DDF) for an application.
FTP	File Transfer Protocol. A protocol used on the Internet for sending files.
HTML Output	A type of eStatement Manager batch job used to create static HTML-formatted email and output files. You publish a DDF file for the Indexer task and a view for this job type.
IIOP	Internet Inter-ORB Protocol. A protocol used for communication between CORBA object request brokers in a TCP/IP environment.
IIS	Internet Information Server. Microsoft's Web server that runs on, and is bundled with, Windows NT and Windows 2000. IIS is limited to the Windows NT platform, but is relatively easy to administer since it is tightly integrated with the operating system. (Netscape's Web servers run on all major platforms, including Windows NT, OS/2, and UNIX.)
Indexer	A type of eStatement Manager batch job used to prepare a data file to be accessed by other production tasks. Indexer extracts meta-data from the application's DDF and stores the content in partitioned database index tables. Indexing an input file enables sub-second statement-retrieval response time across data sets of millions of electronic documents while maintaining the core data in the original input format. You publish a DDF file for the Indexer task, and dynamic Web views which depend on the Indexer job.

Input file (also Data File)	A service provider's file of account or statement data which eStatement Manager extracts and presents on the Web. An input file consists of many individual documents. Also called a source file, eStatement Manager supports a wide variety of print formats, database extracts, and intermediate document composition formats. Each eStatement Manager application must be custom-designed for its unique input file format and content.
ISAPI	Internet Server API. A programming interface on IIS. Using ISAPI function calls, Web pages can run programs written as DLLs on the server, typically to access a database. IIS comes with a DLL that lets embedded queries access ODBC-compliant databases. ISAPI is an alternative to using CGI scripts on Microsoft Web servers. The counterpart to ISAPI on the client side is WinInet.
Java API	Java Application Programming Interface. Prewritten code organized into packages of similar topics. For instance, the Applet and AWT packages include classes for creating fonts, menus, and buttons.
J2EE	Java 2.0 Enterprise Edition
J2EE Application	Any deployable unit of J2EE functionality. This can be a single module or a group of modules packaged into an .ear file with a J2EE application deployment descriptor. J2EE applications are typically engineered to be distributed across multiple computing tiers.
J2EE Server	The runtime portion of a J2EE product. A J2EE server provides Web and/or EJB containers.
J2SE	Java 2.0 Standard Edition
JDBC	Java Database Connectivity. An industry standard for database independent connectivity between the Java platform and a wide range of databases. The JDBC interface provides a call-level API for SQL-based database access.
JDK	Java Development Kit. A software development environment for writing applets and applications in the Java programming language.
JMS	Java Messaging Service. An API for use with enterprise messaging systems such as IBM, MQ Series, TIBCO Rendezvous, etc.
JNDI	Java Naming & Directory Interface. A set of APIs that assists with the interfacing to multiple naming and directory services.
JNI	Java Native Interface. A programming interface (API) in Sun's Java Virtual Machine used for calling native platform elements such as GUI routines. RNI (Raw Native Interface) is the JNI counterpart in Microsoft's Java Virtual Machine.
Job	An eStatement Manager production function designed to produce a particular type of output in an application. All the different types of jobs defined for an application contribute to the process of extracting and presenting cyclical account data for viewing online. Jobs can be scheduled to run automatically on specific days, times, and intervals. See also **Job Type**.
Job Type	The particular function or purpose of an eStatement Manager job. Job types include Indexer, Email Notification, Purge App, Purge Logs, XML Output, HTML Output, and Detail Extractor.

JSP	JavaServer Page. An extensible Web technology that uses template data, custom elements, scripting languages, and server-side Java objects to return dynamic content to a client. Typically the template data is HTML or XML elements, and in many cases the client is a Web browser. (JSPs are converted to Java Servlets by the servlet engine.)
JTA	Java Transaction API.
JTS	Java Transaction Services. A way to maintain control of distributed transactions to protect them from sharing violations or other failures. WebLogic uses JTS internally for its transaction processing functionality, but does not export any of its JTS implementation as a public API.
JVM	Java Virtual Machine. An abstract computing machine, or virtual machine, JVM is a platform-independent programming language that converts Java byte code into machine language and executes it. Most programming languages compile source code directly into machine code that is designed to run on a specific microprocessor architecture or operating system, such as Windows or UNIX. A JVM – a machine within a machine – mimics a real Java processor, enabling Java bytecode to be executed as actions or operating system calls on any processor regardless of the operating system. For example, establishing a socket connection from a workstation to a remote machine involves an operating system call. Since different operating systems handle sockets in different ways, the JVM translates the programming code so that two machines on different platforms can connect.
LDAP	Lightweight Directory Access Protocol. A protocol used to access a directory listing. LDAP support is being implemented in Web browsers and e-mail programs that can query an LDAP-compliant directory. LDAP is expected to provide a common method for searching e-mail addresses on the Internet, eventually leading to a global white pages. LDAP is a sibling protocol to HTTP and FTP and uses the ldap:// prefix in its URL.
Marker	A set of delimiters defining where eStatement Manager can find and extract table or group data from source file. Markers are defined using the DefTool during the application design and development process.
Metadata	Descriptive information about data elements; data about data. In eStatement Manager, metadata is the term used to describe the key data elements used to positively identify a statement within a data source during live retrieval. The metadata is extracted from the input data source and preserved in partitioned index tables during the Indexing job to prepare the data source to be accessed by other production tasks.
MICR	Magnetic Ink Character Recognition, sometime used to refer to the account and transit number at the bottom of a check.
NACHA	National Automated Clearing House Association. The organization that governs and sets the operational rules for all participants of the ACH Network (Term used with Payment.)

NSAPI	Netscape Server API. A programming interface on Netscape's Web Server. Using NSAPI function calls, Web pages can invoke programs on the server, typically to access data in a database. NSAPI is an alternative to using CGI scripts on Netscape Web servers.
OAM	Online account management. The ability to provide live account statement views to customers and manage customer relationships on the Web. Online account management is made available by a service provider. For the customer, OAM can include the ability to make payments online and/or receive statement summaries on a selected payment consolidation portal.
ODBC	Open Database Connectivity. A standard database access method developed by Microsoft to make it possible to access any data from any application, regardless of which DBMS is handling the data. ODBC inserts a middle layer, called a database driver, between an application and the DBMS. The purpose of this layer is to translate the application's data queries into commands the DBMS understands, which requires that both the application and DBMS be ODBC-compliant.
ODFI	Originating Depository Financial Institution. Institution responsible for the origination of ACH transactions; the institution could deposit items directly with an ACH operator. Typically, the ODFI is the online service provider's bank. (Term used with Payment.)
ORB	Object Request Broker. A component in the CORBA programming model, ORB acts as the middleware between clients and servers. In the CORBA model, a client can request a service without knowing what servers are attached to the network. The various ORBs receive the requests, forward them to the appropriate servers, and then send the results back to the client.
Page Style	A definition created in the DDF to identify a statement page (subsequent to the Document Style) that contains a finite subset of the statement's data. A statement can have many pages consisting of one page style or they can have multiple page styles. Examples of different page styles include a summary page, line item detail pages, a cover letter, an ID card, or other types of information formatted differently (and therefore contain separate subsets of data). The page Style definition consists of a name, pattern and the row and column coordinates within which eStatement Manager must search for the start of the data string that differentiates the page Style from others in the statement.
Pattern	The nature and format of a data element to be extracted from an input file. A pattern must be defined for each data element to be extracted from an input file. For example, dollar amounts vary in length (number of digits) across statements, yet have the same general pattern. All addresses vary in length, yet consistently contain both numbers and letters, etc. Patterns are specified in all DDF definitions (page styles, fields, table columns and markers, etc.) as Regular Expressions. To extract a piece of data from an input file, eStatement Manager looks to the DDF file for the extraction rules (the page style, search area and pattern) defined for the data. When eStatement Manager finds a piece of data on that style page, within the defined search area, matching the defined pattern (called pattern matching), it extracts the data.

Payee	A business entity capable of processing or accepting payments. (Term used with Payment.)
PPD	Prearranged Payment or Deposit. Automated consumer payment application by which a consumer can authorize debits or credits to his/her account by a company or financial. Payment supports this type of ACH payment. (Term used with Payment.)
Primary Key	The mandatory field defined on the document style page in the DDF, used to identify the individual or business associated with a statement in the input data source. The primary key is often the account number, however, in the case of statements for recipients with multiple account numbers, another data element must be selected as the Primary Key. Each Primary Key Field instance must be unique to the recipient.
Production	Running eStatement Manager application jobs on a regular basis to provide current account information on demand to Web users.
Publishing	To identify a set of new or updated eStatement Manager design files (typically a DDF, ALF, and HTML templates) for use by an application in a live production environment. The set of files published is called a view. The production process uses the published view files along with input data to enable end-users to view their account information on the Web.
Purge App	The eStatement Manager batch job used to remove an application's index, email, reporting, detail, annotations, and dispute data from the database up to a date specified.
Purge Logs	The eStatement Manager batch job used to remove all rows in the .LOG table up to the date specified.
RDFI	Receiving Depository Financial Institution. The receiving financial institution is an ACH transaction's final destination. Typically, the customers' banks are the RDFIs. (Term used with Payment.)
Regular Expression	In eStatement Manager, a programming expression containing the search pattern, or rules, for locating and extracting a data element from an input file. A regular expression is a syntactical combination of symbols that represent a value, and consists of at least one operand (a value) and can have one or more operators (actions). Successful regular expressions, also known as regexes, take advantage of the known characteristics of a data element, such as the general content and relative position of characters (numbers, decimal points, letters, etc.) while accounting for expected unknowns or variables such as exact length. A regular expression can be fairly general or more specific, depending on how exact the pattern needs to be to reliably extract the correct data.
RMI	Java Remote Method Invocation. A distributed object model for Java program to Java program, in which the methods of remote objects written in the Java programming language can be invoked from other Java virtual machines, possibly on different hosts.

Search Area	The area on a document page in an input data source, identified by start and end coordinates of rows and columns, where eStatement Manager looks for the start of a particular data element for identification (page styles and markers) or extraction (fields and table columns). The search area for each field must be defined in the DDF during application development. It is not possible to identify the precise location where data starts and ends in an input file, but it is possible to identify an area, relative to the page, where a data element can be found, along with the pattern of data expected.
Service Provider	Business or other entity that provides online account management using one or more eStatement applications.
Source file	See **Input file**.
SWIFT	Society for Worldwide Interbank Financial Transfers. (Term used with Payment.)
Table	A group of related data elements, defined as columns, which could iterate within the statement. Since table data is extracted together from the input data source, it is formatted and presentment together as well. Tables are generally built to extract and present iterative or transactional information, such as telephone or credit card transaction detail. Tables can also be built for multi-row data elements, such as customer addresses and messages. Tables are built using DefTool, and consist of a table anchor, marker pair, and one or more columns.
Table Anchor	A required element of a table, which, when detected during extraction of the table, confirms that data exists on relevant rows. The table anchor can be an internal anchor (a column) or an external anchor (a data string that resides on every row where there is table data and is not extracted as part of the table). See also **Anchor**.
User ID	The identification a Web-user enters to log into an eStatement application.
Version Set	A set of design files (typically a DDF/ALF pair and all statement and group HTML templates) published for use in data processing and live retrieval of statement views on a service provider's online account management site. A version set is a dated instance of a view. A new version sets can be published when the view design files are updated for ongoing presentment and marketing purpose.
View	A set of design files (typically a DDF/ALF pair and all statement and group HTML templates) which result in a particular Web view of a document. An application can have multiple views for different presentment purposes, such as generating dynamic or static HTML-formatted output and email notification. Different views can also be created to show different levels of information in a document, such as statement summary information, statement detail, etc. See also **Version Set**.
View Name	The name identifying a particular application view.
View Type	The type of output, or view, a version set produces: HTML, CSV, XML, Chart, XSLT, or XML Query.

VM	Virtual Machine. A self-contained operating environment that behaves as if it is a separate computer. For example, Java applets run in a Java Virtual Machine that has no access to the host operating system. Advantages to this design include system independence and security.
Volume	A data input file that has been successfully processed by the Indexer job and referenced in the volumes table. A date processed value in the Date Accepted or the Date Rejected column indicates the volume status (primarily intended for systems using an internal verification process).
XML	Extensible Markup Language. An open standard for identifying data from the W3C. XML is used to define data elements on Web pages and B2B documents. It uses a similar tag structure as HTML; however, HTML defines *how* elements are displayed and XML defines *what* those elements contain. HTML uses predefined tags, but XML allows tags to be defined by the page developer. XML can identify virtually any data items, such as product, sales rep and amount due, enabling Web pages to function like database records. By providing a common method for identifying data, XML supports B2B transactions and is expected to become the dominant format for electronic data interchange. XML is a subset of SGML (Standard Generalized Markup Language), while HTML is an application of SGML.
XML Output	A type of eStatement Manager batch job used to generate XML-formatted email and output files. You publish a DDF file with an XML Output job (for use by both the Indexer and XMLFormatter tasks).
XSL	Extensible Style sheet Language. A style sheet format for XML documents. It is the XML counterpart to the Cascading Style Sheets (CSS) language in HTML, although XML supports CSS1 and CSS2 as well. XSLT (XSL Transformations) are extensions to XSL for converting XML documents into XML or other document types and can be used independently of XSL.

8 Index