Administration Guide

Telco e-Billing Manager (Consumer Edition)

V4.5
Document ID: TBMG-04-4.5-01
Data Published: 7.31.2004

edocs, Inc., One Apple Hill Dr., Natick, MA 01760

Printed in the USA.

Table of Contents

Preface

About Customer Self-Service and edocs' Tools

edocs' Telecommunications Applications Suite includes every application that communications service providers need to enable a complete online customer-self service experience at their website. The suite includes software applications for:

- e-Billing and Payment
- Service and Order Management
- Point-of-Sale
- Reporting and Analytics
- Rate Plan Advice

edocs' Customer Self-Service applications for the telecommunications industry combine edocs' unrivaled Customer Self-Service and e-Billing software suite with its extensive industry domain expertise. The packaged, out-of-the-box applications are tailored to solve communications service providers' distinct business problems and to meet communications industry-specific process requirements.

edocs' Telecommunications Applications Suite includes:

Telco e-Billing Manager

edocs' Telco e-Billing Manager is a complete e-billing application for communications service providers that gives business and consumer customers valuable and convenient access to their communications bills along with the ability to easily make online payments.

Telco Service Manager – Business Edition

edocs' Telco Service Manager – Business Edition enables business customers of communications service providers to manage every aspect of their service relationship online. From a single convenient interface, customers can easily activate and manage subscriptions, order new products and services, and report and resolve problems for individual employees, as well as company departments and divisions, across the entire organization.

Telco Service Manager – Consumer Edition

edocs Telco Service Manager – Consumer Edition enables consumer customers to activate and manage service, buy new products and services, resolve problems and manage their own accounts online using virtually any communications device.

Telco Service Manager – Channel Edition

edocs' Telco Service Manager – Channel Edition delivers self-service capabilities to communications dealers and point-of-sale locations, allowing them to improve the effectiveness of the retail sales channel by quickly and easily activating customers online, selling new products and services, and responding to service requests.

Telco Analytics Manager

edocs' Telco Analytics Manager is a reporting solution for business customers that empowers both individual employees and business managers to analyze and understand their communications costs and usage by investigating and identifying trends and patterns across multiple views of their own unique organization.

Rate Plan Advisor

edocs' Rate Plan Advisor is a web-based application that recommends the ideal rate plan for wireless subscribers in real-time. Individual consumers as well as large businesses can analyze their actual historical voice/mobile/data usage, find the best-fit rate plans, and compare the features offered by those plans. With its intuitive wizard user interface, Rate Plan Advisor quickly guides end-customers or customer service representatives through the entire analysis process. In addition, a service provider's customer care and marketing groups can also use Rate Plan Advisor to identify pre-churn subscribers, simulate new rate plans, and run predictive analytics.

About This Guide

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

- Set up a new TBM application and the associated jobs using the TBM Command Center.
- Produce regular online statements electronically and manage the ongoing live production process.

This guide assumes you have:

- Installed Telco e-Billing Manager in your J2EE environment.
- Designed and developed the necessary application files (using DefTool and Composer in a Windows 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 TBM Command Center:

Online Help	How to Access
Comprehensive Command Center Help	Click **`Help`** on the Command Center menu. Help contains additional information about administering 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 administration Help.
A PDF version of this guide	A PDF version of this document is located on your product CD.

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

Print Document	Description
Telco e-Billing Manager Installation Guide	How to install TBM and configure it in a distributed environment.
Telco e-Billing Manager Data Definition Guide	How to create Data Definition Files (DDFs) for use in indexing your application and extracting data for live presentment.
Telco e-Billing Manager Presentation Design Guide	How to create Application Logic Files (ALFs) to present statement data for dynamic online display.

Obtaining edocs Software and Documentation

You can download edocs software and documentation directly from Customer Central at https://support.edocs.com. After you log in, click on the Downloads button on the left. When the next page appears, you will see a table displaying all of the available downloads. To search for specific items, select the Version and/or Category and click the Search Downloads button. If you download software, an email from edocs Technical Support will automatically be sent to you (the registered owner) with your license key information.

If you received an edocs product installation CD, load it on your system and navigate from its root directory to the folder where the software installer resides for your operating system. You can run the installer from that location, or you can copy it to your file system and run it from there. The product documentation included with your CD is in the Documentation folder located in the root directory. The license key information for the products on the CD is included with the package materials shipped with the CD.

If You Need Help

Technical Support is available to customers who have an active maintenance and support contract with edocs. Technical Support engineers can help you install, configure, and maintain your edocs application.

This guide contains general troubleshooting guidelines intended to empower you to resolve problems on your own. If you are still unable to identify and correct an issue, contact Technical Support for assistance.

Information to provide

Before contacting edocs Technical Support, try resolving the problem yourself using the information provided in this guide. If you cannot resolve the issue on your own, be sure to gather the following information and have it handy when you contact technical support. This will enable your edocs support engineer to more quickly assess your problem and get you back up and running more quickly.

Please be prepared to provide Technical Support the following information:

Contact information:

- Your name and role in your organization.
- Your company's name
- Your phone number and best times to call you
- Your e-mail address

Product and platform:

- In which edocs product did the problem occur?
- What version of the product do you have?
- What is your operating system version? RDBMS? Other platform information?

Specific details about your problem:

- Did your system crash or hang?
- What system activity was taking place when the problem occurred?
- Did the system generate a screen error message? If so, please send us that message. (Type the error text or press the Print Screen button and paste the screen into your email.)

- Did the system write information to a log? If so, please send us that file. For more information, see the *TBM Troubleshooting Guide*.
- How did the system respond to the error?
- What steps have you taken to attempt to resolve the problem?
- What other information would we need to have (supporting data files, steps we'd need to take) to replicate the problem or error?
- **Problem severity:**
- Clearly communicate the impact of the case (Severity I, II, III, IV) as well as the Priority (Urgent, High, Medium, Low, No Rush).
- Specify whether the problem occurred in a production or test environment.

Contacting edocs Technical Support

You can contact Technical Support online, by email, or by telephone.

edocs provides global Technical Support services from the following Support Centers:

US Support Center
Natick, MA
Mon-Fri 8:30am – 8:00pm US EST
Telephone: 508-652-8400

Europe Support Center
London, United Kingdom
Mon-Fri 9:00am – 5:00 GMT
Telephone: +44 20 8956 2673

Asia Pac Rim Support Center
Melbourne, Australia
Mon-Fri 9:00am – 5:00pm AU
Telephone: +61 3 9909 7301

Customer Central
https://support.edocs.com

Email Support
mailto:support@edocs.com

Escalation process

edocs managerial escalation ensures that critical problems are properly managed through resolution including aligning proper resources and providing notification and frequent status reports to the client.

edocs escalation process has two tiers:

1. **Technical Escalation -** edocs technical escalation chain ensures access to the right technical resources to determine the best course of action.

2. **Managerial Escalation** - All severity 1 cases are immediately brought to the attention of the Technical Support Manager, who can align the necessary resources for resolution. Our escalation process ensures that critical problems are properly managed to resolution, and that clients as well as edocs executive management receive notification and frequent status reports.

By separating their tasks, the technical resources remain 100% focused on resolving the problem while the Support Manager handles communication and status.

To escalate your case, ask the Technical Support Engineer to:

1. Raise the severity level classification.

2. Put you in contact with the Technical Support Escalation Manager.

3. Request that the Director of Technical Support arrange a conference call with the Vice President of Services.

4. Contact VP of Services directly if you are still in need of more immediate assistance.

1

Overview of the TBM Application Setup Process

Before Getting Started

Your Telco e-Billing Manager (TBM) application was created by a project team in your organization. A TBM application consists of various design files used in a live production environment with your TBM 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 TBM GUI design tools, DefTool and Composer, to create these files.

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

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

- Become familiar with the TBM 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.

The Application Setup Process

The process of setting up a new application in the TBM 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 Telco e-Billing Manager application, you must:

1. **Create a new application.** This short step requires you to define, or name, the application in the TBM 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. See ”What Jobs Do I Need to Create?” on page 14 for a description of 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 TBM 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 TBM 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 17.

Once you have defined your TBM 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 TBM 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 “Bulk Publishing*”on page 70* for details.

What Jobs Do I Need to Create?

Each TBM 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 may need to create and configure a combination of 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.
- **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.

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

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 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 (also called a "hit list").

The Indexer job consists of separate tasks that run sequentially; see "Creating and Configuring an Indexer Job" on Page 27 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 how to set up an EmailNotification job, see “Creating and Configuring an EmailNotification Job”on Page 34.

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 details on setting up a Purge App job, see “Creating and Configuring a Purge App Job” on Page 40.

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 details on setting up a Purge Logs job, see “Creating and Configuring a Purge Logs Job” on page 45.

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 47.

For details on setting up an XML Output job, see “Creating and Configuring an XML Output Job” on Page 52.

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 TBM.

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 57.

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.

A TBM 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, TBM 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 edocs 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 edocs 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 Composer.

In addition to HTML-formatted views, TBM application views can dynamically present data in one of the 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, which 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. TBM uses these stylesheets to identify and format the type of data the user requests on-demand.

When setting up a new TBM application, you make the dynamic Web view files available to an application by "publishing" them using the edocs' Publisher tool. See "What Files Do I Need to Publish?"on page 22.

Publish application files in bulk to save time moving an application between development, testing, and production servers.

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.)

Sample TBM application

Your TBM 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 *opt/edocs/TBM/eStatement/samples/NatlWireless* directory (where *opt/edocs/TBM* is your *EDX_HOME*, the default directory where you installed TBM).

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

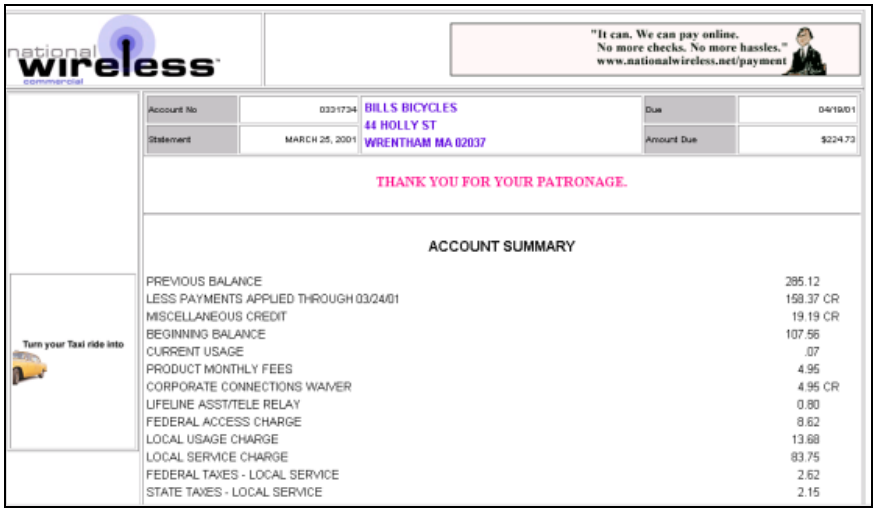
Sample dynamic HTML views

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*

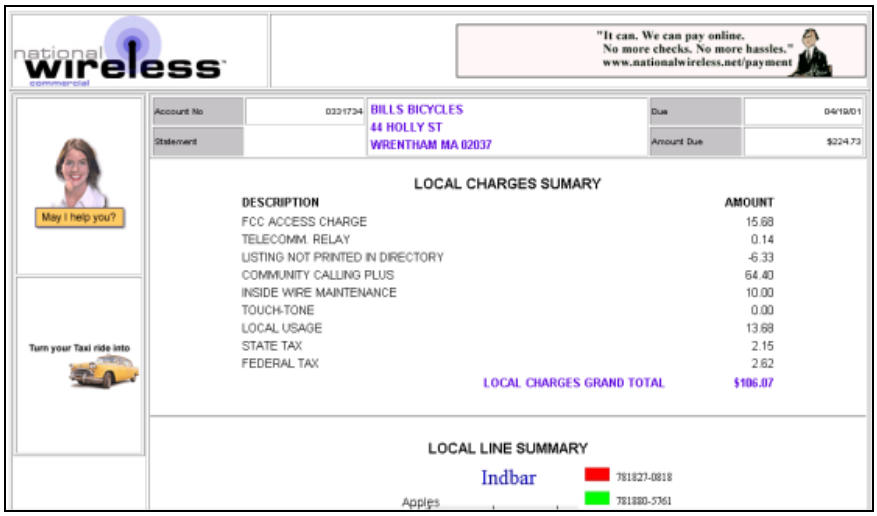
This view presents the primary page of the NatlWireless application:



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 *TBM Deploying and Customizing J2EE Applications* guide.

Email Notification views

TBM 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

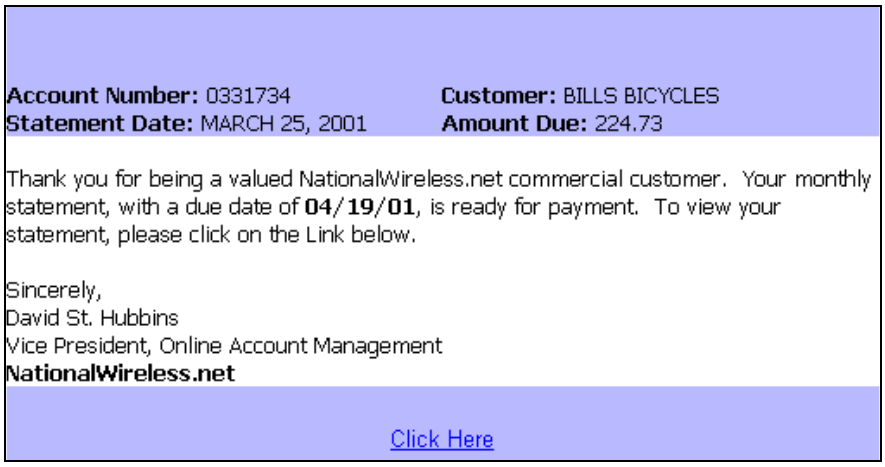
The NatlWireless sample application contains a sample email notification view, consisting of the following files, to generate a sample email for telecommunication customers:

- *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 TBM 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.

TBM 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.

Publish application files in bulk to save time moving an application between development, testing, and production servers.

For information about publishing dynamic Web views (version sets) individually, see "How to Publish Your Application's Dynamic Web Views" on page 66.

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 TBM 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 TBM 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

Dynamic Web view format	Files you publish in a version set for this view format:
CSV	DDF TOK
XML	DDF
Chart	A Properties file
XSLT	XSL DDF
XML Query	XML

See "How to Publish Your Application's Dynamic Web Views" on Page 66 for instructions.

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 file next to each version set in the search results. To view the rest of the contents in a readme.txt, use a text editor or see "Viewing the Contents of a Design File" on page 76.

For consistency, it is a good idea to establish a standard format for the type of information your organization includes in readme.txt files.

You can publish a readme.txt for each version set when publishing individually or in bulk.

To create and publish a readme.txt for a version set:

1. Manually create a file called readme.txt.
2. On the first line, enter a one-line description of the version set that you want to appear in Publisher searches. End this line with a paragraph mark (Windows) or with a /R or /N (Unix).
3. You can add additional lines of information to the file, however only the first line appears in Publisher.
4. Publish the readme.txt when you publish the associated version set. When publishing version sets individually, you can browse and select the readme.txt along with the version set files. Include the readme.txt in the view name folder when bulk publishing.

2

Setting Up a New Application and Jobs

Logging Into the Telco e-Billing 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.

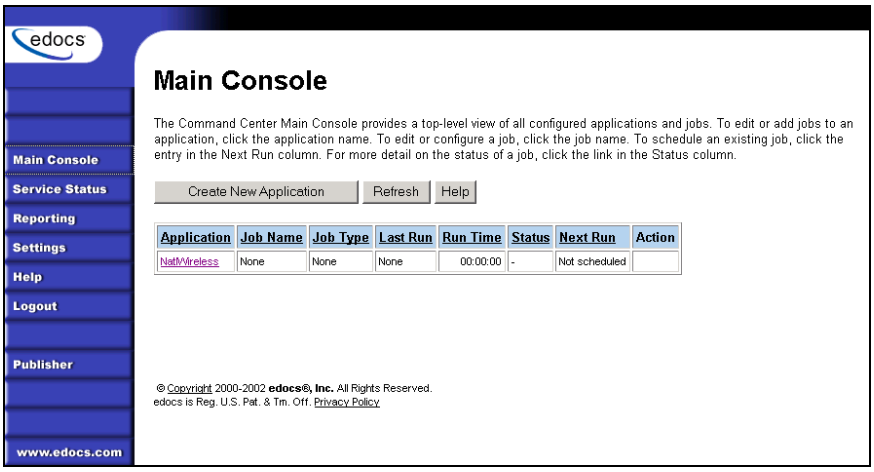
The Command Center is a secure application that requires you to log in with an administrator's ID and password. If you forget the Command Center password, contact your system administrator or the person who installed TBM.

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

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 Command Center servlet configured when TBM was installed, such as `http://dusky:7001/edocs`.
4. On the Login Administrator page, enter the administrator's ID and password. The default ID is **`admin`** and the password is **`edocs`**.

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



To log out of the Command Center:

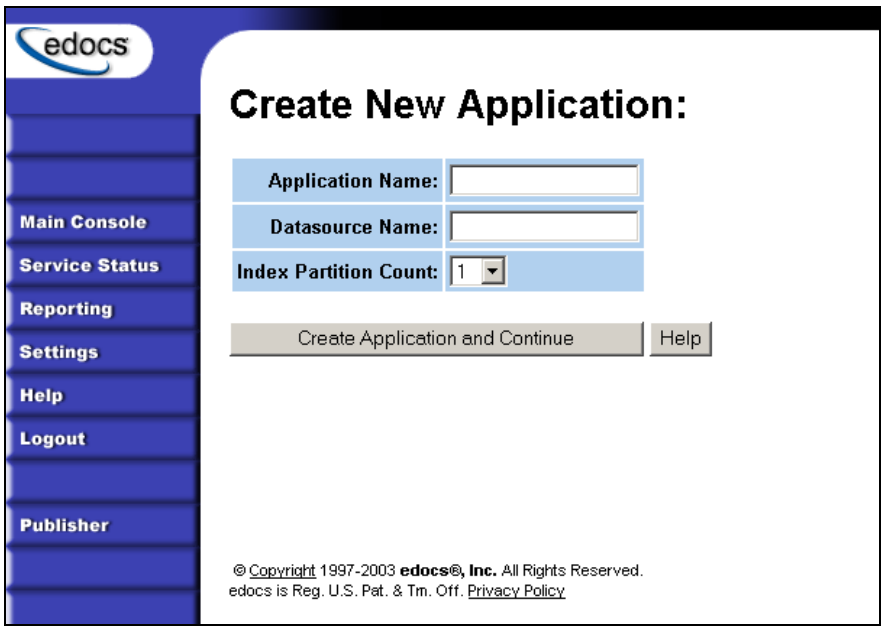
- Click **`Logout`** on the Main Console.

Creating a New Application

To create a new application:

1. Go to the TBM Command Center.

   Click **`Create New Application`** from the Main Console. TBM 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 TBM 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 TBM. 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`**. TBM 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 TBM application (DDN) you create in the Command Center. When creating a TBM 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.

For information on developing a custom datasource EJB, please consult your edocs 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.

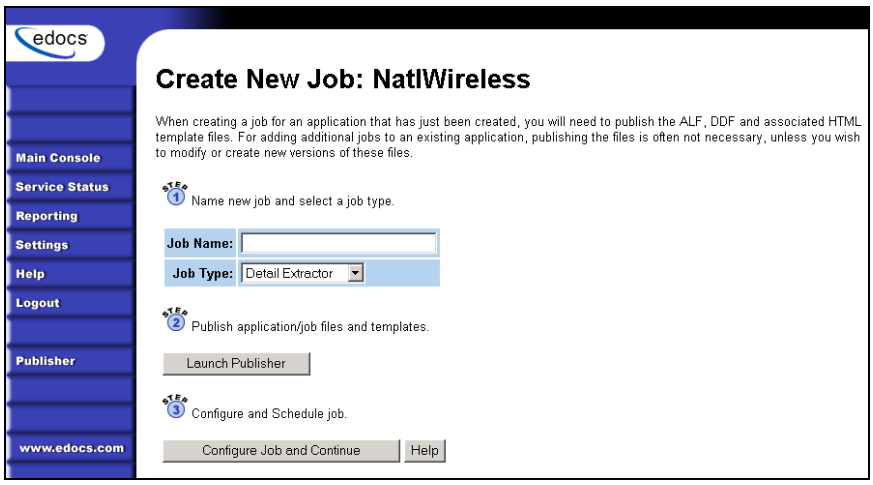
Each time you run an Indexer job, it looks for multiple data files in the input directory and processes them one at a time.

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.

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. TBM displays the Create New Job screen:

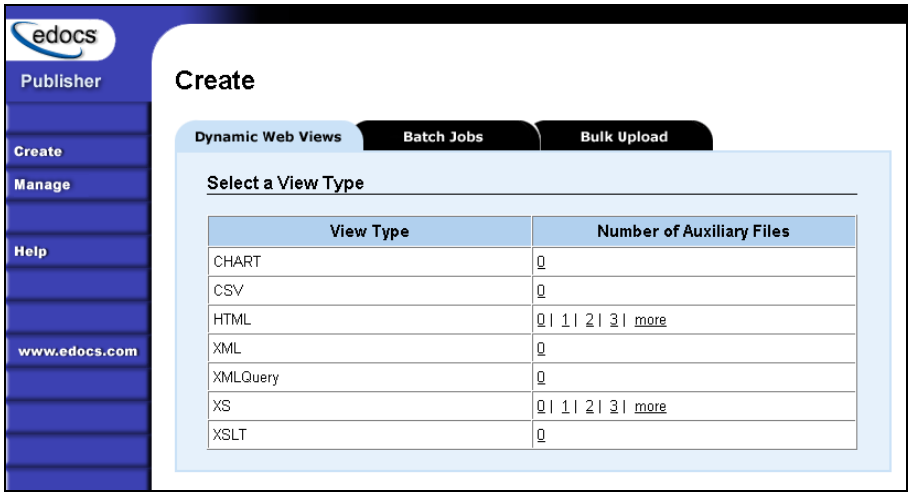


2. Enter a meaningful name for the Indexer job. The job name can contain alphanumerics and underscores, but no spaces. The Indexer job name cannot start with a numeric.

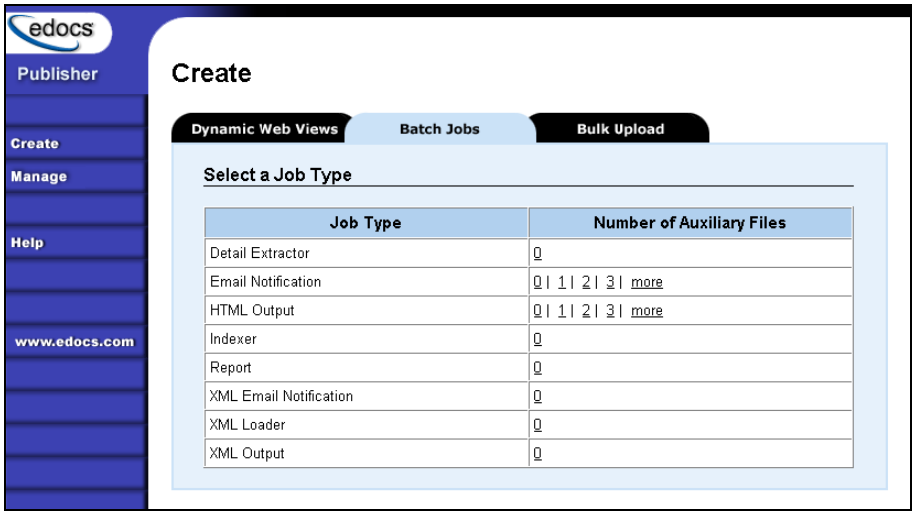
3. Select the Indexer job type from the drop-down menu.

4. If you plan to publish this application's files in bulk, skip to Step 13. Otherwise, click **`Launch Publisher`**.

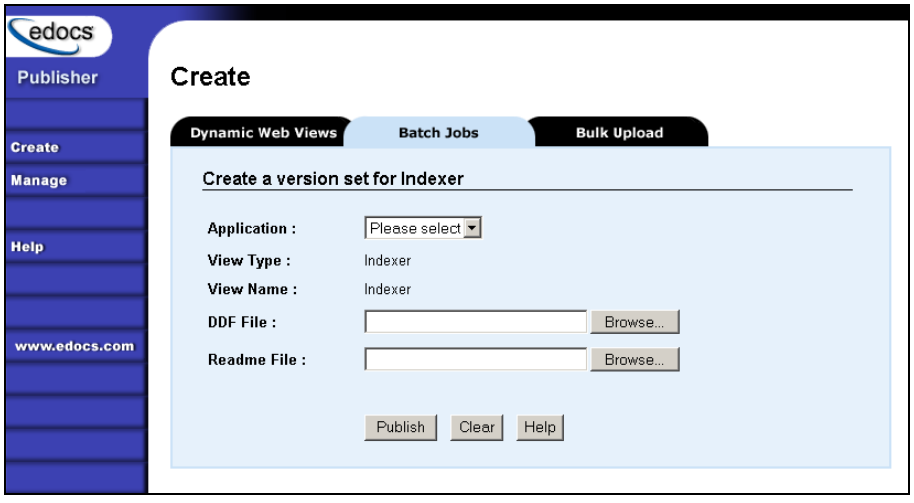
5. Click **`Create`** on the Publisher Menu.



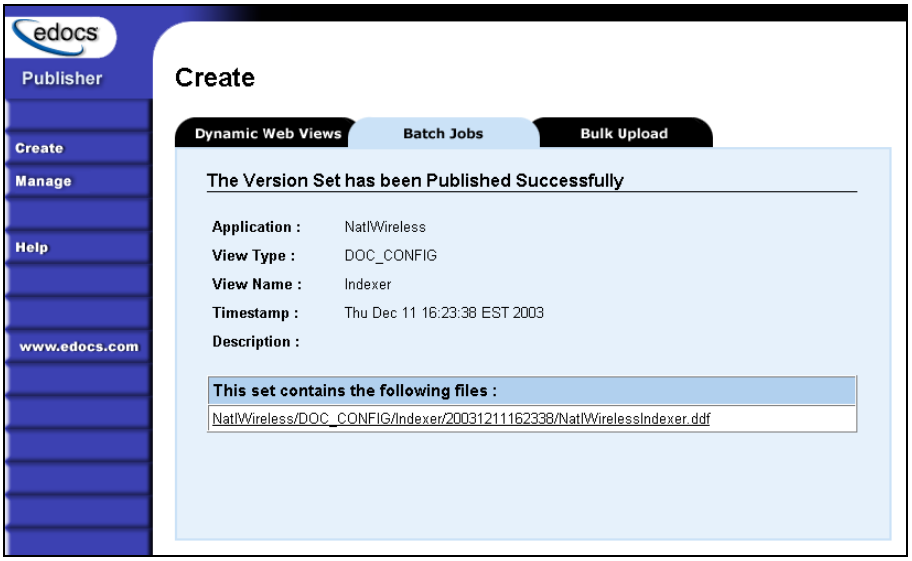
6. Click the **`Batch Jobs`** tab.



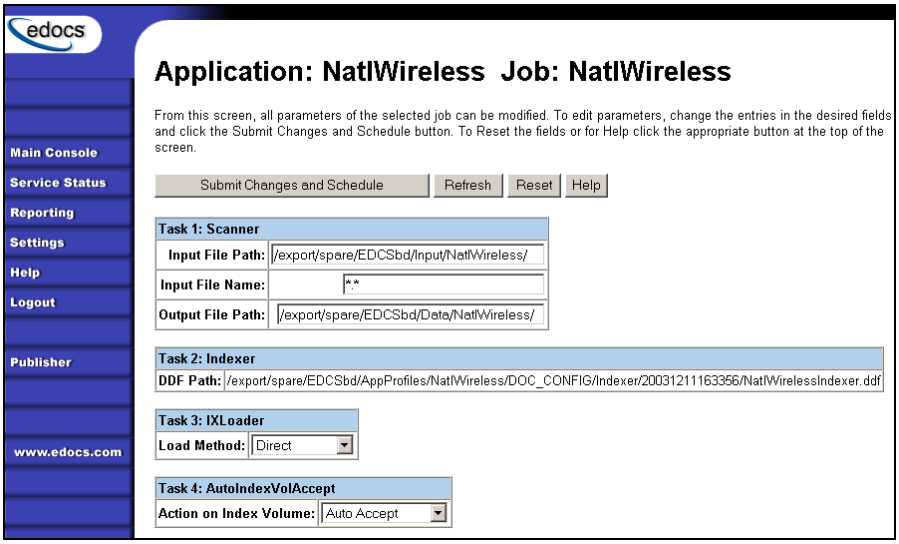
7. For job type Indexer, click **`0`** (Number of Auxiliary files). Publisher displays the Create a Version Set For Indexer screen:



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).
11. Click **`Publish`**. Publisher lets you know when it has successfully published the version set and displays details about the DDF:



12. Close the Publisher window.
13. At the Create New Job screen in Command Center, click **`Configure Job and Continue`**. TBM 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`**. TBM asks "OK to submit this configuration?"

16. Click **`OK`**. TBM 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 *EDX_HOME/eStatement/Input/Application Name* (where *EDX_HOME* is the default directory where you installed TBM and *Application Name* is the name of your TBM 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 *EDX_HOME/eStatement/Data/Application Name* (where *EDX_HOME* is the default directory where you installed TBM and *Application Name* is the name of your TBM 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. edocs 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. edocs 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 TBM 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:

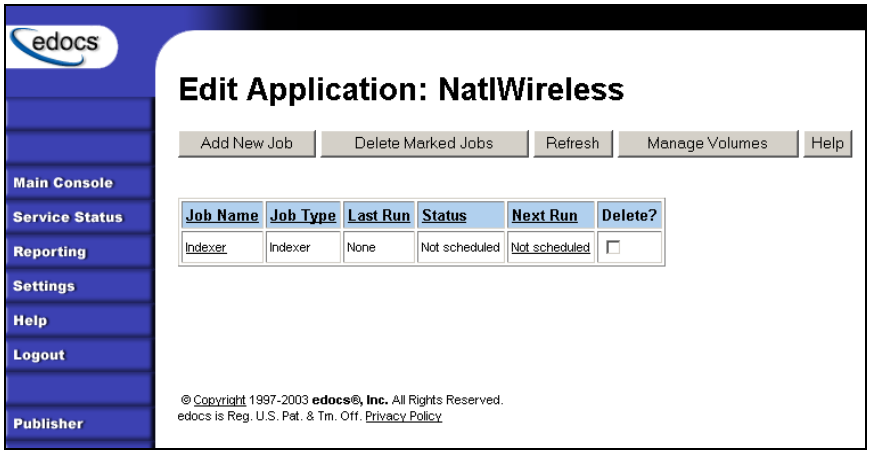
- Specify configuration settings for the two production tasks that run sequentially as part of the job: IVNScanner and MailNotification
- Publish the email view files (DDF, ALF, and HTML templates) created for the intended email message

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

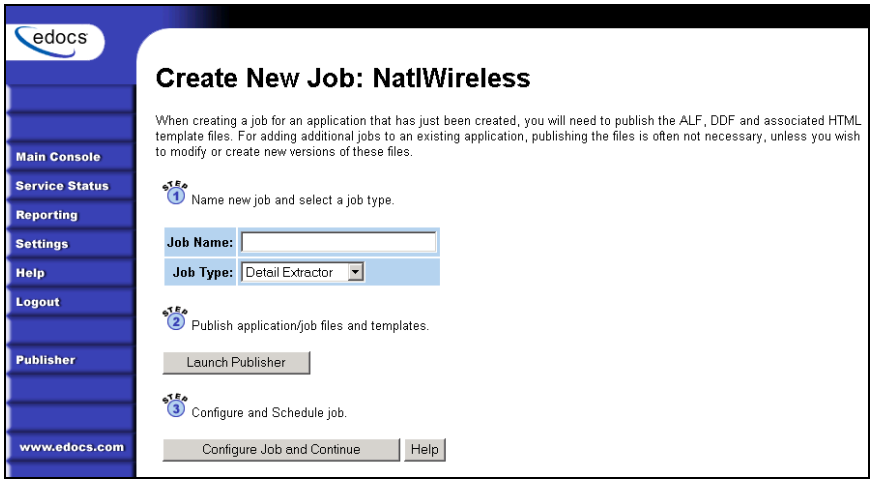
You can publish the EmailNotification view files (DDF, ALF, and HTML's) at the time you create and configure the EmailNotification job or you can publish these and all other required application files in bulk. Bulk publishing makes it easier to move application files between development, testing, and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

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`**. TBM displays the Create New Job screen:

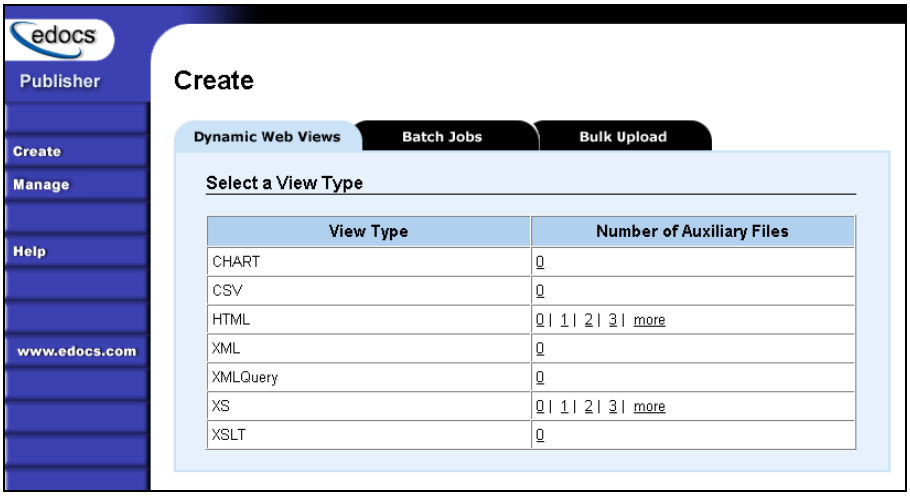


3. Enter a meaningful name for the job. The job name must start with an alpha character. The rest of the characters can be alphanumeric and can contain underscores, but no spaces.

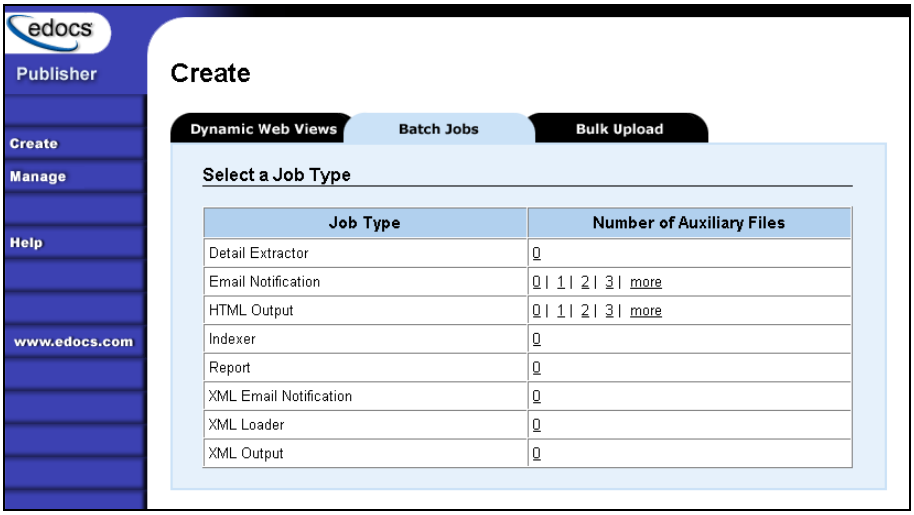
4. Select the Email Notification job type from the drop-down menu.

5. If you plan to publish this application's files in bulk, skip to Step 13. Otherwise click **`Launch Publisher`**.

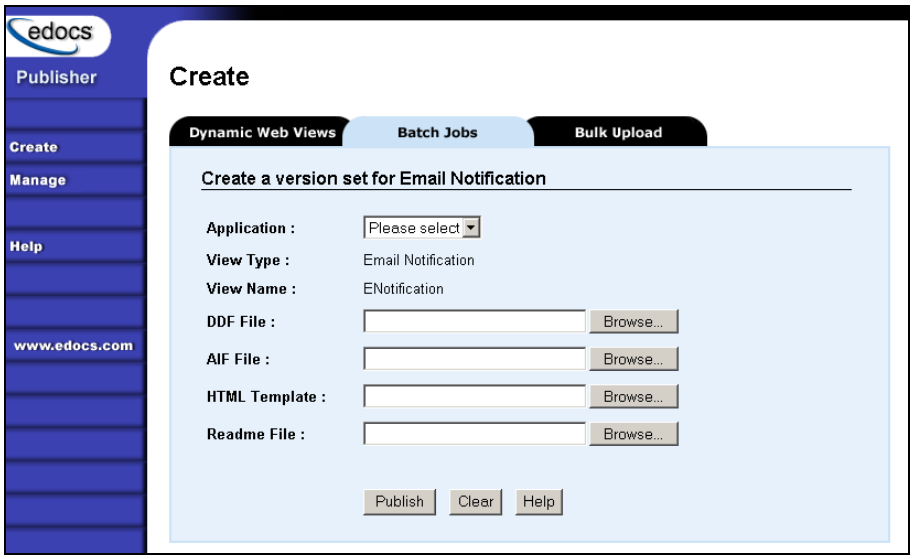
6. Click **`Create`** on the Publisher Menu.



7. Click the **`Batch Jobs`** tab.



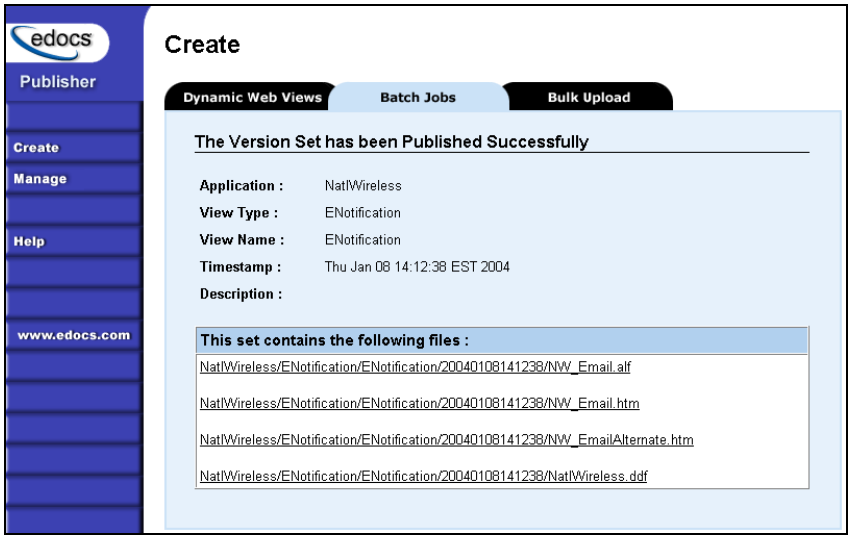
8. For job type Email Notification, click the number of auxiliary HTML templates the view uses. (The number of HTML templates *in addition to* the default template. If the view uses one HTML file, click **`0`**, if it uses two HTML templates, click **`1`**, etc.) Publisher displays the Create a Version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



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

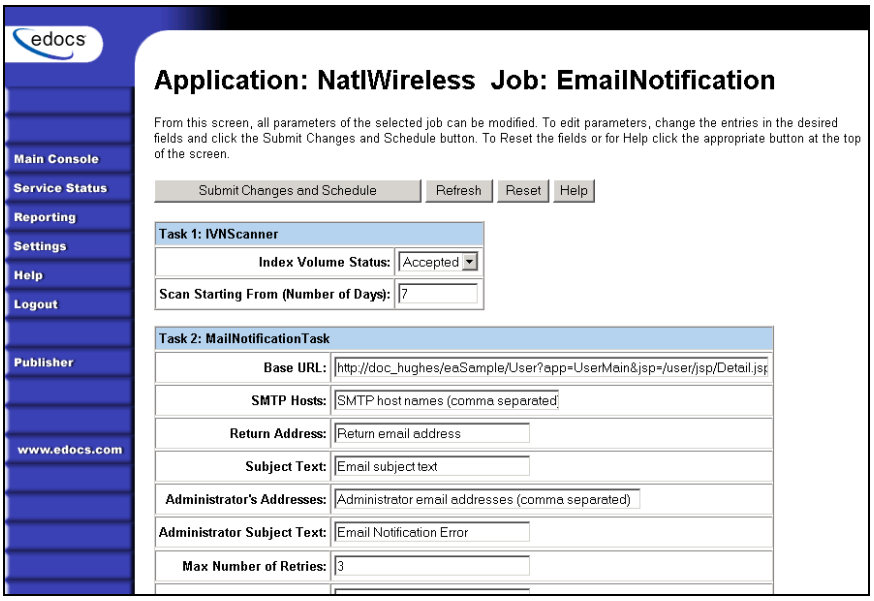
10. Browse and select the readme.txt for this version set (if you've created one).

11. Click **`Publish`**. Publisher lets you know when it has successfully published the version set 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`**. TBM 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`**. TBM submits the job configuration parameters and displays the Schedule screen. You can schedule the EmailNotification job later. 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 TBM handles delivery errors.

You also specify enrollment model settings to tell TBM where to look for email enrollment information. If you use TBM enrollment, by default, the system uses the email addresses found in the TBM database. 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 TBM 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 TBM should retry sending email if it fails. The default is 60 minutes.
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 edocs enrollment applications (edocs or CDA). Leave this field blank if you do not have customizations to either edocs 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 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.

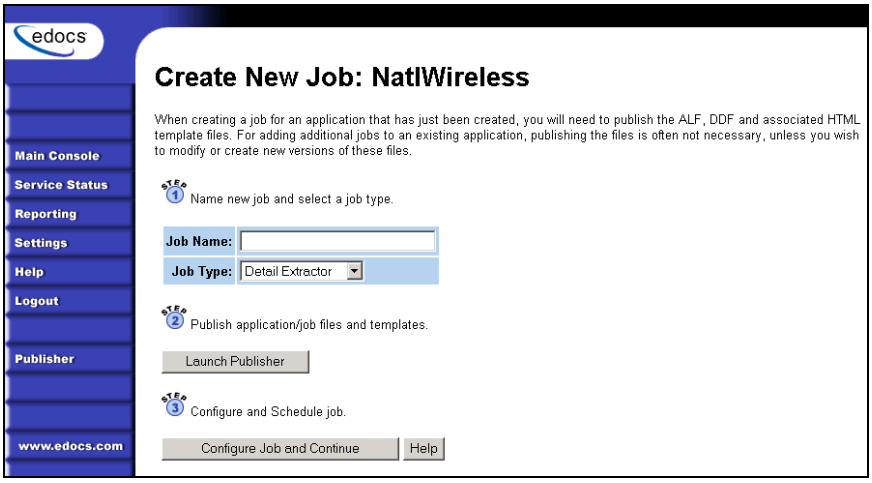
When you configure a Purge App job, you specify settings for the four production tasks that run sequentially as part of the job:

- 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`**. TBM 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`**. TBM 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`**. TBM 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 Created or 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**
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 you 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.

PurgeActivityData Task Configuration	
Field	**What to enter/select**
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.

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. Run this job to free 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.

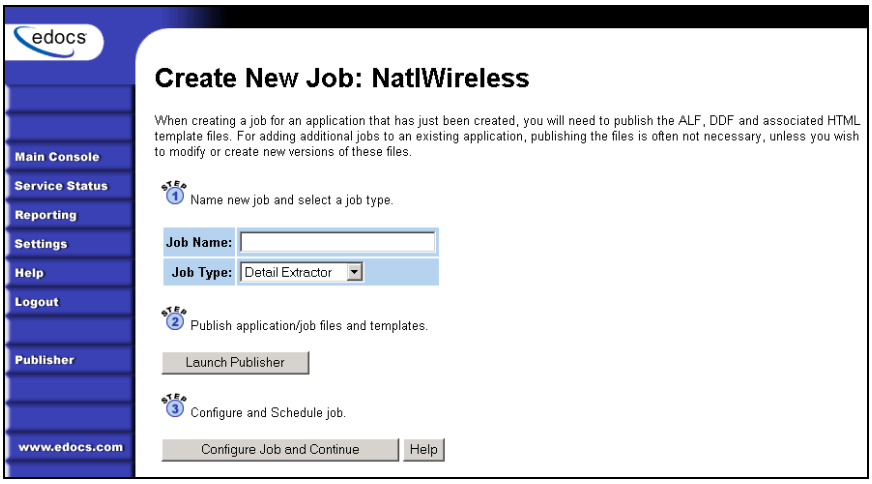
Purge Logs removes data globally (for all applications you have).

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.

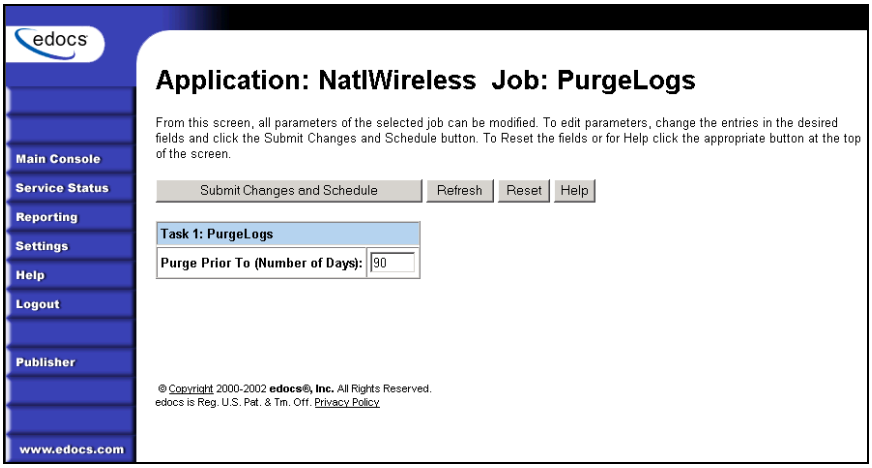
2. Click **`Add New Job`**. TBM displays the Create New Job screen:



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`**. TBM displays the Purge Logs configuration screen:



6. Specify the configuration parameters for the PurgeLogs task.

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

8. Click **`Main Console`**.

Task 1: 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.

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.

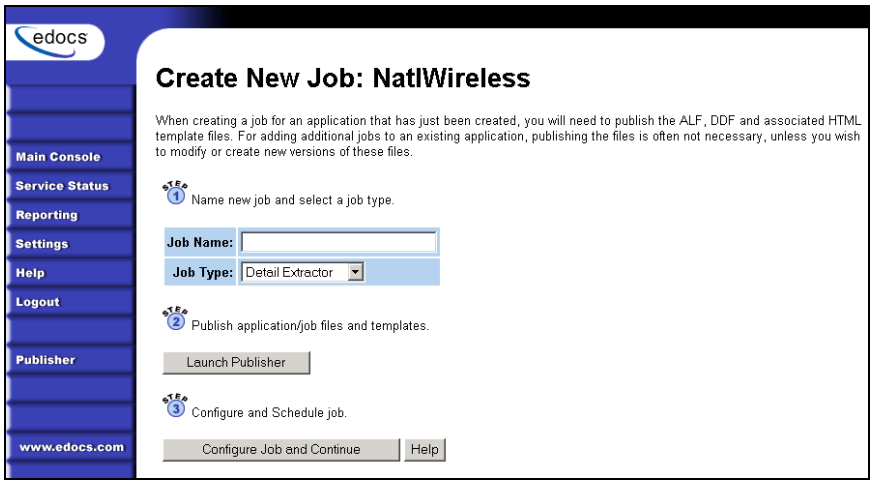
Review the task and field configuration settings in this section to determine which options to set for your application.

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.

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`**. TBM displays the Create New Job screen:

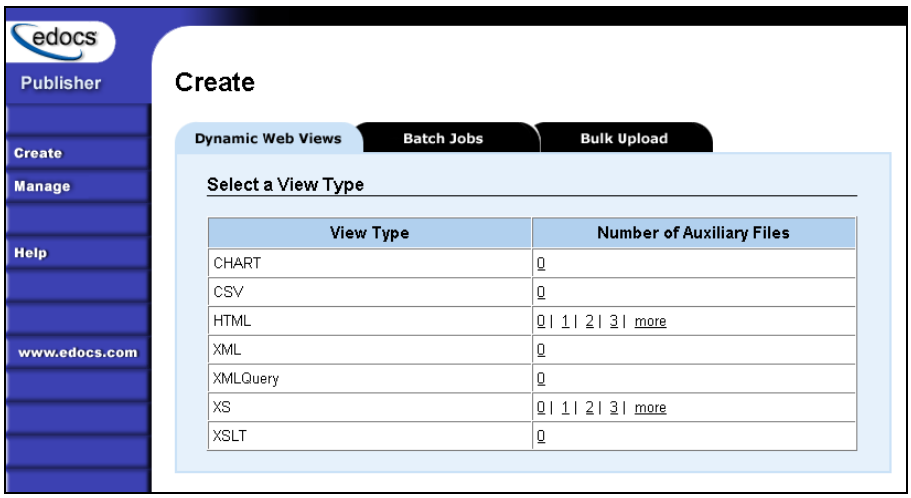


3. Enter a meaningful name for the HTML Output job. The job name must start with an alpha character. The rest of the name can be alphanumeric and can have underscores, but no spaces.

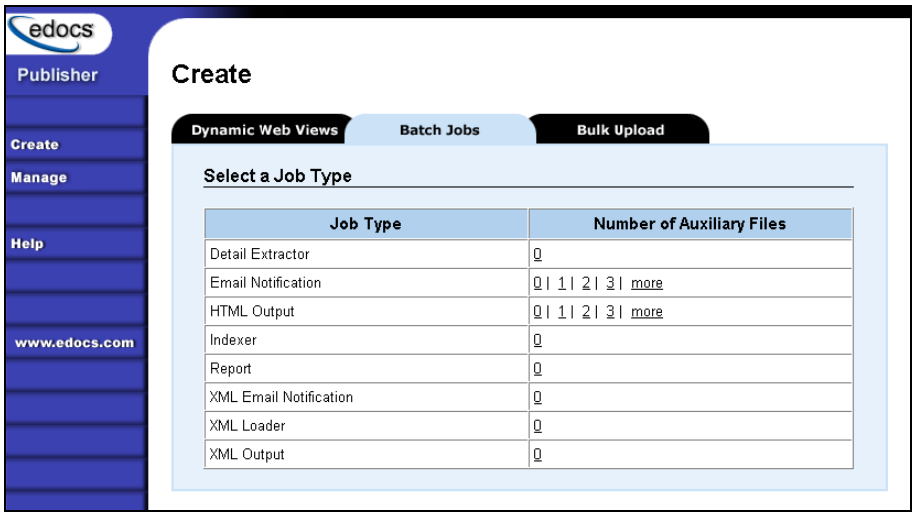
4. Select job type HTML Output.

5. If you plan to publish this application's files in bulk, skip to Step 17. Otherwise click **`Launch Publisher`**.

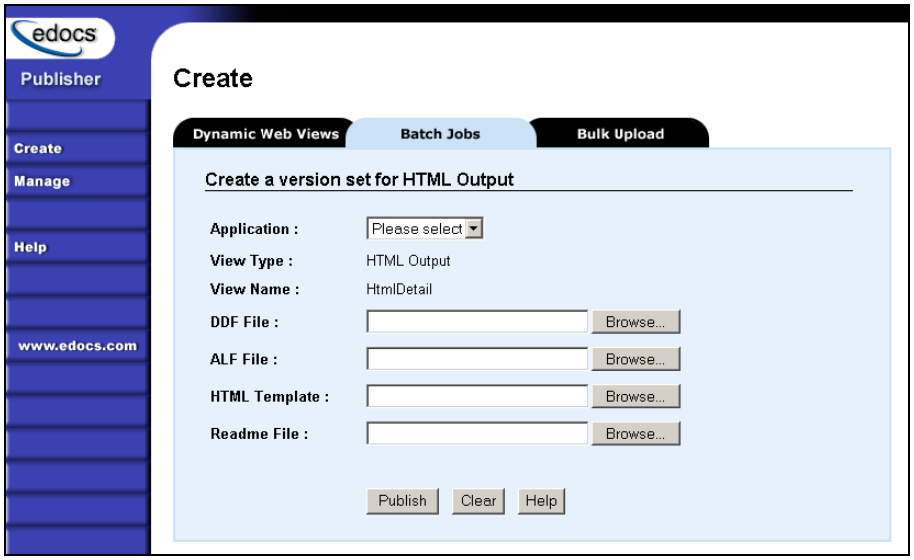
6. Click **`Create`** on the Publisher Menu.



7. Click the **`Batch Jobs`** tab.

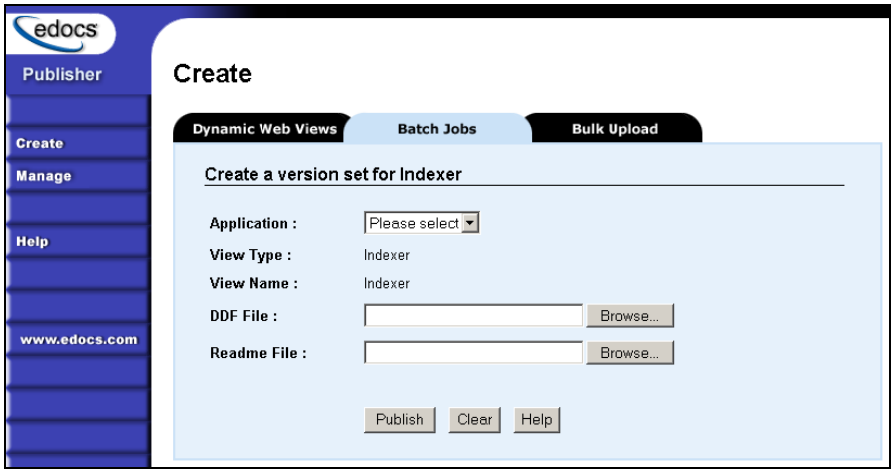


8. For job type HTML Output, click the number of auxiliary HTML templates the view uses (the number *in addition to* the default template). If the view uses one HTML file, click **0**, if it uses two HTML templates, click **1**, etc.) Publisher displays the Create a Version Set for HTML Output screen:



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).
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.) TBM displays the Create a Version Set for Indexer screen:

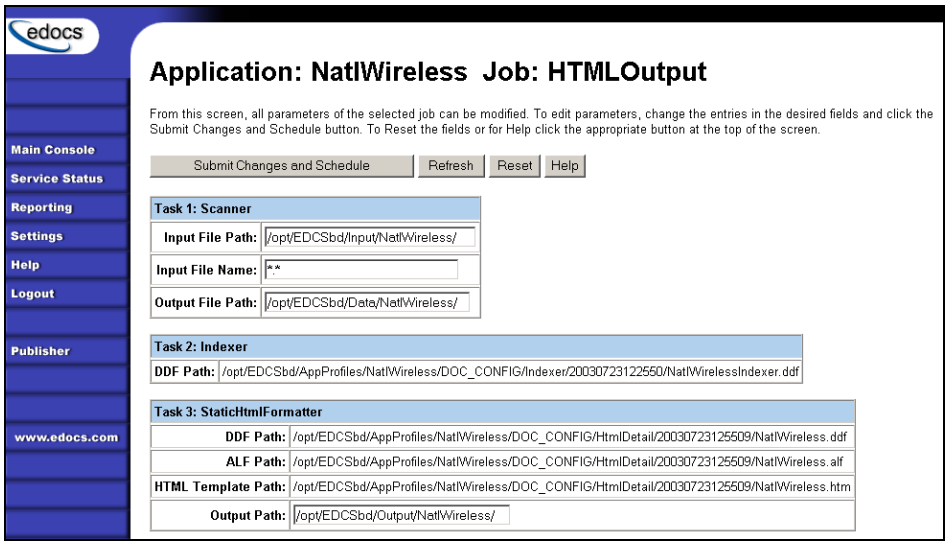


14. Specify the DDF file for the Indexer task.

15. Click **`Publish`**. TBM 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`**. TBM 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`**. TBM 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 31 for a description of Scanner task parameters.

Task 2: Indexer

See page 31 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 TBM to place the Static HTML-formatted output. The default directory is *EDX_HOME/eStatement/Output/Application Name* (where *EDX_HOME* is the default directory where you installed TBM, and *Application Name* is the name of your TBM 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.

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.

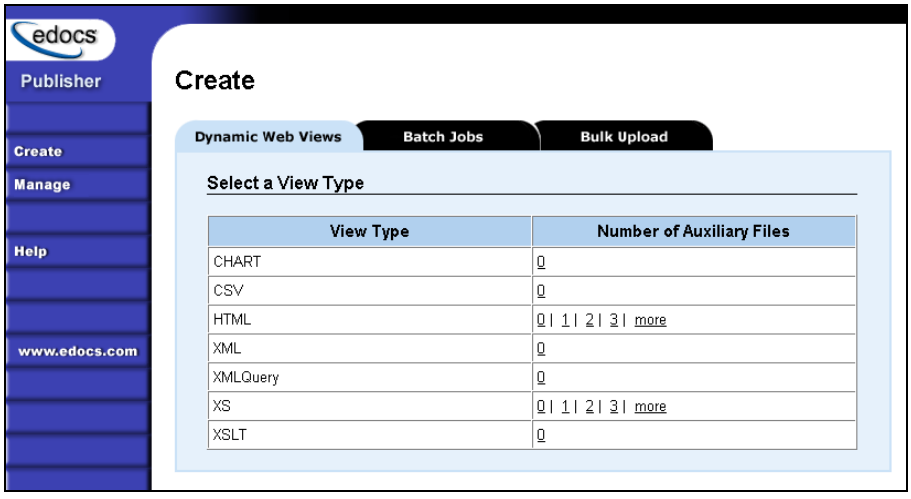
Review the task and field configuration settings in this section to determine which options to set for your application.

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

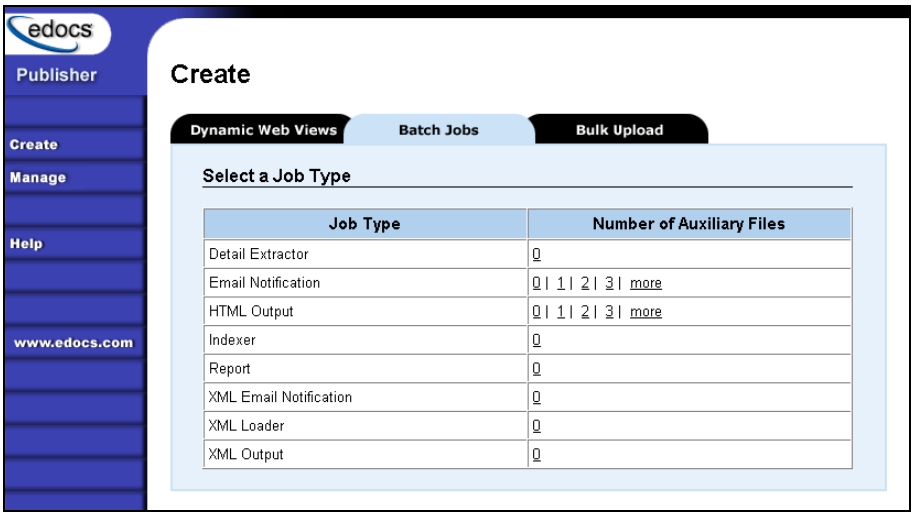
Tip You can publish the required DDF file at the time you create and configure the XML Output job or you can publish it and all other required application files in bulk. Bulk publishing makes it easier to move application files between development, testing, and production servers.

To create and configure an XML Output job:

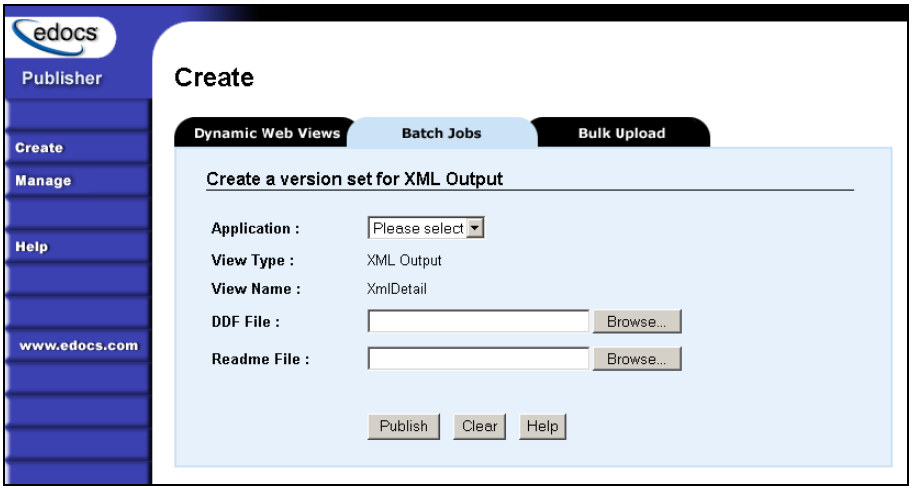
1. On the Main Console, click the application name, listed under Applications in the table. The Edit Application screen appears.
2. Click **`Add New Job`**.
3. Enter a meaningful name for the XML Output job. The job name must start with an alpha character. The rest of the name can be alphanumeric and 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`**.
6. Click **`Create`** on the Publisher Menu.



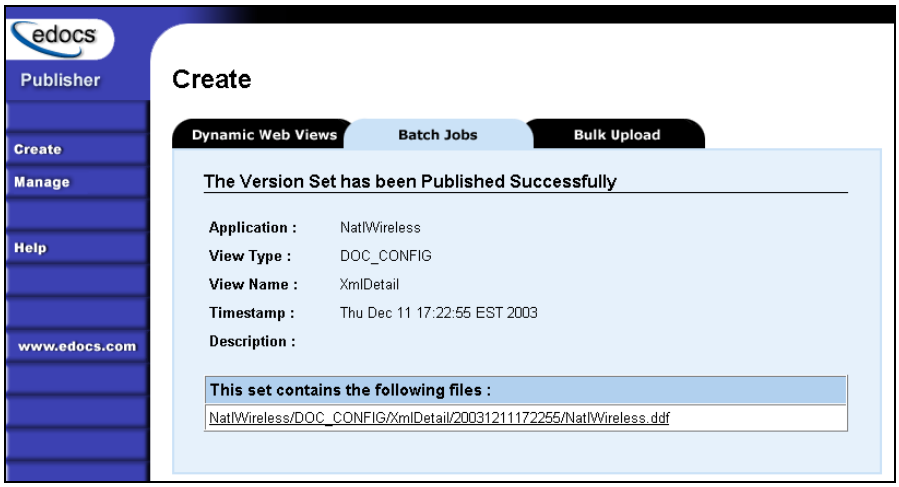
7. Click the **`Batch Jobs`** tab.



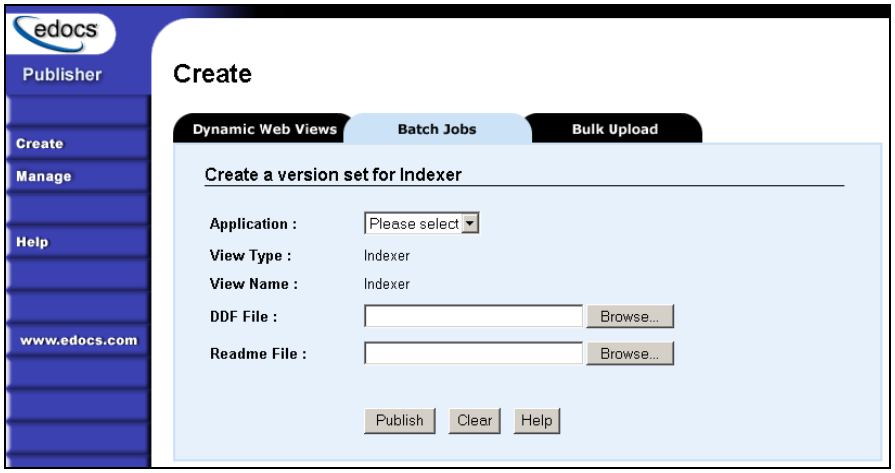
8. For the job type XML Output, click `0` (Number of Auxiliary files). Publisher displays the Create a Version Set for XML Output screen:



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).
12. Click **`Publish`**. Publisher lets you know when it has successfully published the version set and displays details about the DDF file:



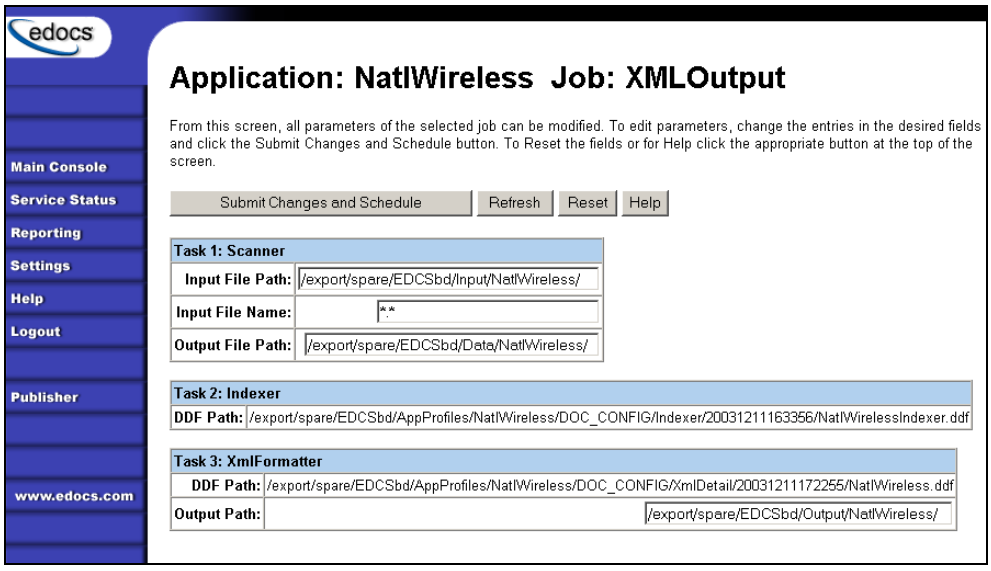
13. If you *don't* already have an Indexer job configured for this application, click the **`Batch Jobs`** tab and publish a DDF file for the Indexer task that runs as part of the XML Output job. (If you already have an Indexer job for this application, go to Step 16.) TBM displays the Create a Version Set for Indexer screen:



14. Select the DDF file to use for the Indexer task and click **`Publish`**. TBM 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`**. TBM 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`**. TBM 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 31 for a description of Scanner task parameters.

Task 2: Indexer

See page 31 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 *EDX_HOME/estatement/Output/Application Name* (where *EDX_HOME* is the default directory where you installed TBM, and *Application Name* is the name of your TBM 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 TBM to place the Static XML-formatted output. The default directory is *EDX_HOME/estatement/Output/Application Name* (where *EDX_HOME* is the default directory where you installed TBM, and *Application Name* is the name of your TBM 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 TBM 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.

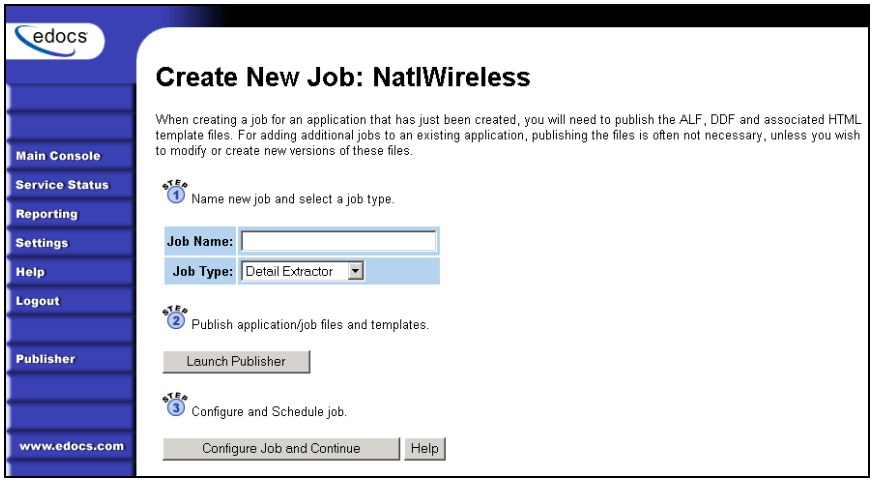
Review the task and field configuration settings in this section to determine which options to set for the Detail Extractor job.

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.

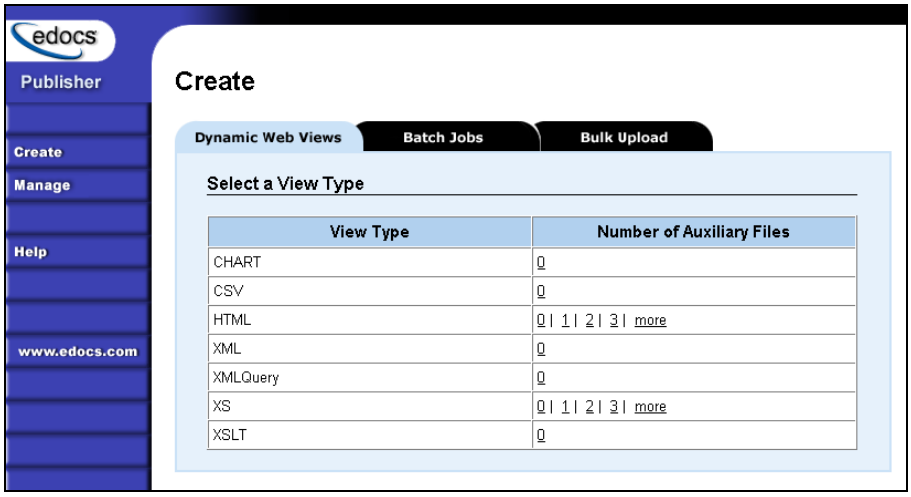
To create and configure a Detail Extractor job:

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`**. TBM displays the Create New Job screen.

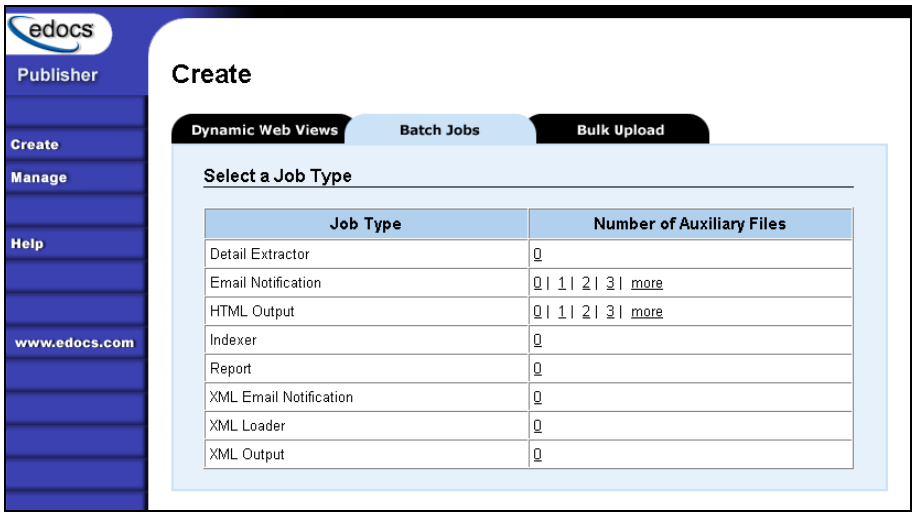


3. Enter a meaningful name for the Detail Extractor job. The job name must start with an alpha character. The rest of the name can be alphanumeric and can have underscores but no spaces. (Using the name of the table defined in the DDF for the job name can help you remember which data the job uploads.)
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`**.

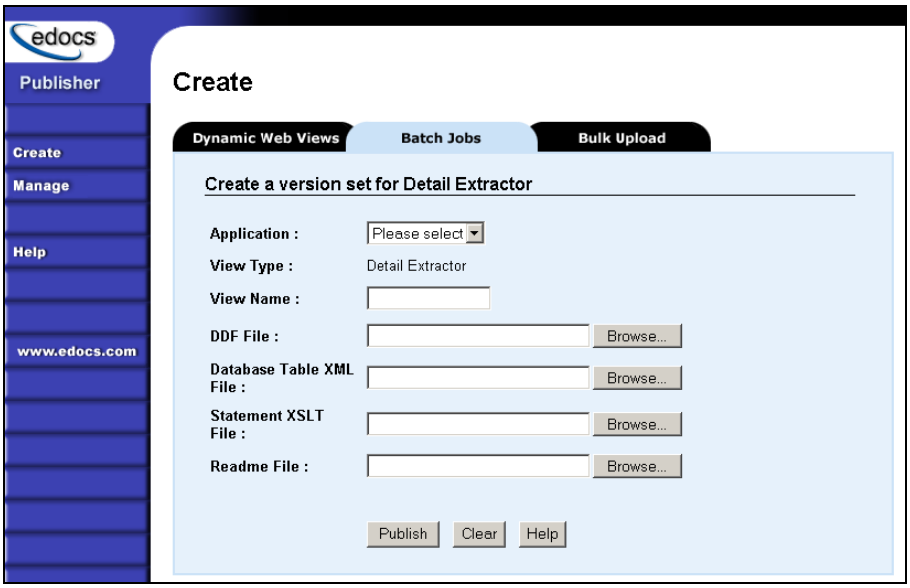
6. Click **Create** on the Publisher Menu.



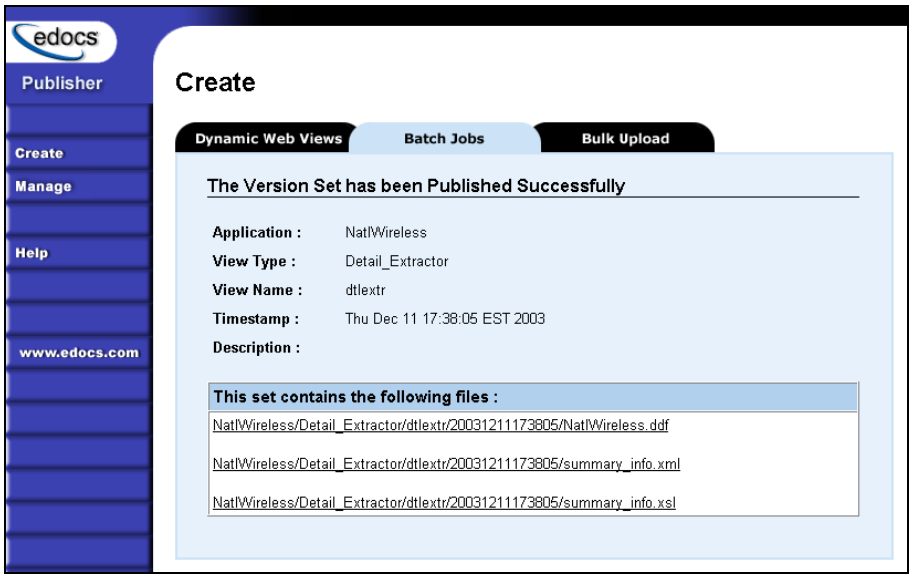
7. Click the **Batch Jobs** tab.



8. For the job type Detail Extractor, click **0** (Number of Auxiliary files). Publisher displays the Create a Version Set for Detail Extractor screen:

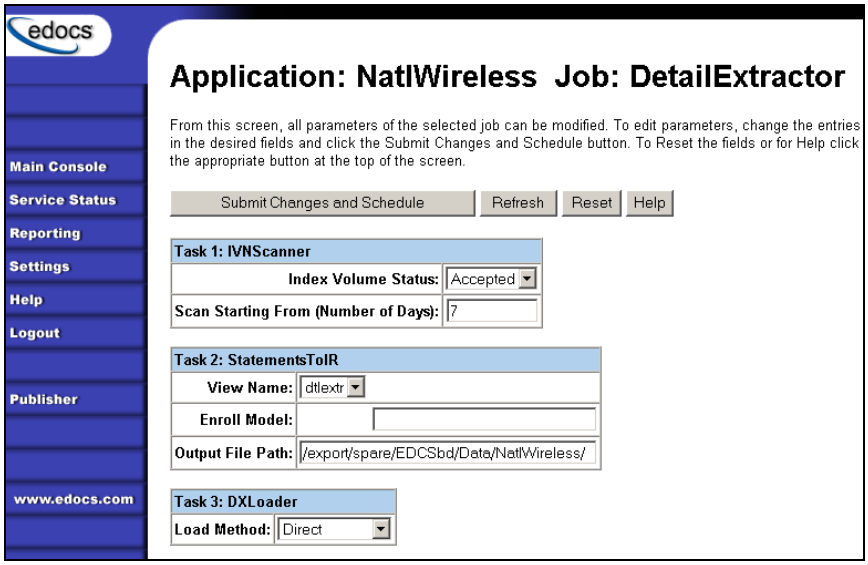


9. Select the application from the drop-down list.
10. Specify a view name for the job that uniquely identifies the view of table data the job extracts. (Using the name of the table defined in the DDF for the view name can help you remember which view maps to which table.) Do not reuse the name of a 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).
15. Click **`Publish`**. Publisher lets you know when it has successfully published the version set 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`**. TBM 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`**. TBM 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. edocs 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. edocs recommends this method for DB2.

3

Publishing and Using Version Sets

When to Publish New Version Sets

As part of the initial process of setting up your 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 presents a user with an online view of a statement. "Publishing" a version set using the edocs 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 DDF.

When you publish a new version set for the view, the new version set is used 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. You can also bulk publish multiple version sets for one or more applications at a time.

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.

Each time you publish a new version set, edocs:

- 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

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.

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

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 "How to Publish Your Application's Dynamic Web Views" on page 66 for instructions.

How to Publish 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.

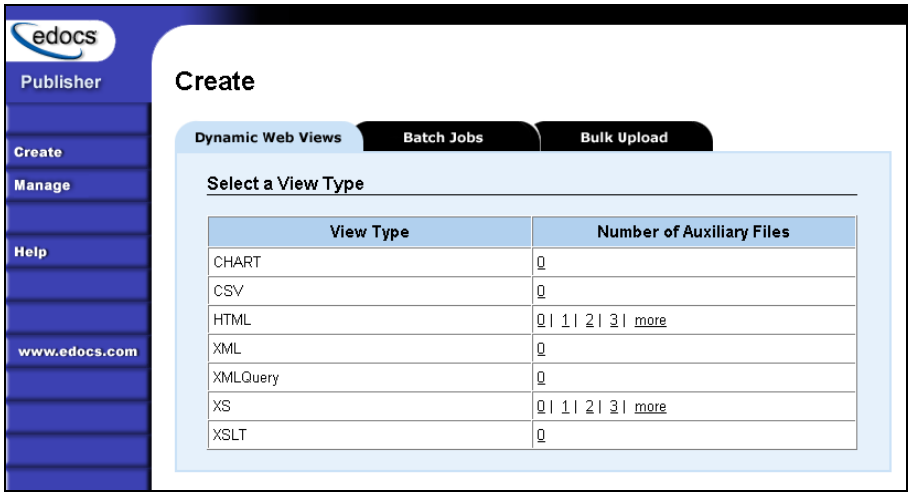
You can publish dynamic Web views individually or in bulk. Bulk publishing makes it easier to move application files between development, testing, and production servers.

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.

To publish a single dynamic Web view for an application:

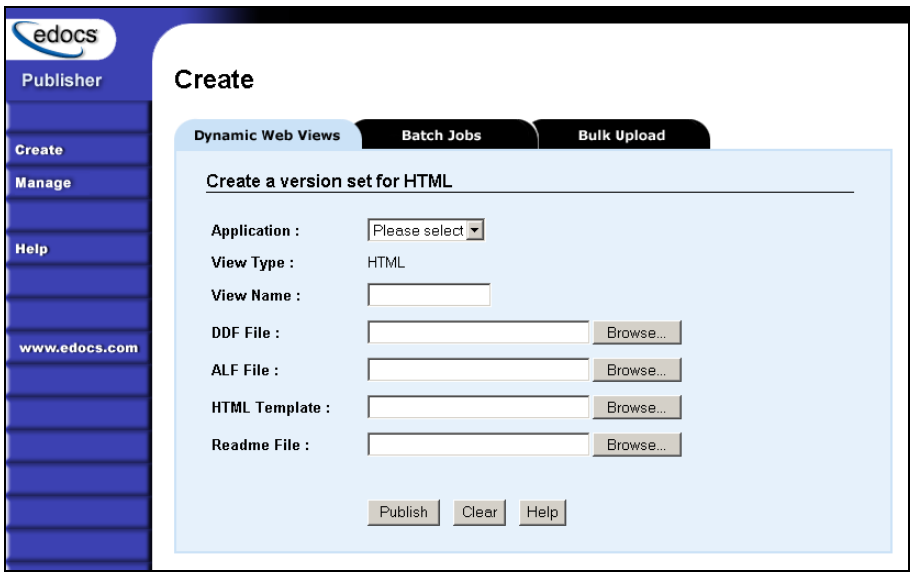
1. Click **`Publisher`** on the Command Center Main Console. TBM launches Publisher.

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).

8. Click **`Publish`**. Publisher displays the Submission screen showing the files you published:

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.

Downloading Application Files for Bulk Publishing

To publish files in bulk, you must first create an archive file containing the application files - version sets for dynamic Web display and batch jobs. If the files are already published on the source server, you can use Publisher to automatically download complete applications or individual version sets to an archive file. (You also have the option to create an archive file manually; see "Archive File Format for Bulk Publishing" on page 70.

Once you create an archive file, you must then upload (publish) the archive to the target server.

To download one or more applications for bulk publishing:

1. Running Command Center on the source server, click Publisher from the Main Console.
2. Click **`Manage`** on the Publisher menu.
3. Click the **`Bulk Download`** tab. Publisher lists all the applications it finds on the connected server.
4. Select the individual applications you want to download to an archive file or click the check box next to "Applications" to select all.
5. Click Download Selected. Publisher displays a file download warning.
6. Click **`Save`** to save the application files to an archive file. Publisher displays the Save As dialog.
7. Select a target location where you want to save the archive. Also specify an archive file name and type, either .zip or .jar. Publisher downloads all version sets found in the application directories for each selected application on the current server into one compressed (.zip or .jar) file on the selected location. Note that Publisher downloads only the most recent copy of each version set. When finished, Publisher displays the Download Complete dialog.

To downloading individual application version sets:

You can download individual version sets for an application using the Publisher search feature.

By specifying search filtering criteria, you can identify and generate the list of version sets, view a list of the files that make up any of the version sets listed, view their content if you want, then choose which version sets to download to a .zip or .jar archive file right from there.

Bulk Publishing

When setting up a new application or updating an existing one, you can publish the dynamic Web view version sets and batch job files one at a time or all together, in "bulk." Bulk publishing lets you move application files between development, test, and production environments all at once, saving time and minimizing errors.

To publish in bulk, you must create an archive file containing all the files you want to publish. If the files have been published on the source server, you can use Publisher to automatically download the files for one or more applications to an archive file, which you can then upload (bulk publish) to the target server. Publisher also lets you download individual version sets to an archive.

You also have the option to create the archive .zip or .jar manually.

Publishing adds to any existing views and view types already there for the application on the target server.

To publish all files for one or more applications in bulk, you must:

1. Use Publisher to download application files to a *.zip* or *.jar* archive file or manually place application files in an archive file, making sure to organize them in the proper file folder structure.
2. Be sure the applications (DDNs) are defined on the target server before uploading.
3. Run Publisher on the target server and upload, or publish, the files from the *.zip* or *.jar* archive.

Archive File Format for Bulk Publishing

To publish application version set files in bulk, you must create a .zip or .jar compressed archive file containing the application files you want to publish. You can then upload this file to a target server, saving the time of publishing individual version sets using the Publisher interface. You can create the archive file automatically using Publisher, or manually using the format described here.

To create an archive file manually, you must place all application files in the proper file folder structure in a .zip or .jar file so that Publisher can later identify and the files and upload them to the existing applications. When uploading this archive file, Publisher performs extensive validations and will fail if the .zip or .jar application files are not in the proper folder structure, named properly, etc.

Place all application files in a .zip or .jar file in the following folder structure:

Product-Specific Name/Application Name/View Type/View Name/Application Files

For example, you would store the dynamic HTML statement view files for NatlWireless in the following folder structure:

AppProfiles/NatlWireless/HTML/HtmlDetail/NatlWireless.ddf

AppProfiles/NatlWireless/HTML/HtmlDetail/NatlWireless.alf

AppProfiles/NatlWireless/HTML/HtmlDetail/NatlWireless.htm

AppProfiles/NatlWireless/HTML/HtmlDetail/readme.txt

Name the folders according to the following view types and names for each batch job and dynamic Web view type you want to include in the bulk publish:

Batch Job Type	View Type	View Name
Detail Extractor	Detail_Extractor	*User-defined*
EmailNotification	ENotification	ENotification
HTML Output	DOC_CONFIG	HtmlDetail
Indexer	DOC_CONFIG	Indexer
Report	DOC_CONFIG	Report
XML Email Notification	XENotification	XENotification
XML Loader	DOC_CONFIG	XMLLoader
XML Output	DOC_CONFIG	XmlDetail

Dynamic Web View Type	View Type	View Name
CHART	CHART	*User-defined*
CSV	CSV	*User-defined*
HTML	HTML	*User-defined*
XML	XML	*User-defined*
XMLQuery	XMLQuery	*User-defined*
XS	XS	*User-defined*
XSLT	XSLT	*User-defined*

Warning: Be sure to include the files defined for a particular version set and required for a view. The upload will fail if it finds an incomplete version set.

Uploading an Archive File for Bulk Publishing

To publish files in bulk, you must download application files to a compressed format (*.zip* or *.jar*) archive file or manually create the archive file, then upload the archive to a target server. The archive file must conform to the folder structure required by Publisher.

When you upload an archive file, Publisher performs extensive checks on the file and its contents. Publisher checks the archive for: valid file format; required folder structure and naming conventions for the DDN, view types, view names, and file names; the presence of each required file type and auxiliary files for each particular view type; minimum and maximum number of files allowed in a view type; and the existence of the specific files referenced within each view. Publisher also checks each file format against its schema, and checks the target server for availability, sufficient space, and for the existence of the application DDN's there.

To upload (publish) application files in bulk:

1. Be sure that the applications you are uploading are defined on the target server.
2. Running Command Center on the target server, click **`Publisher`** from the Main Console.
3. Click **`Create`** on the Publisher menu.
4. Click the **`Bulk Upload`** tab.
5. Click **`Browse`** to select the archive file you want to upload.
6. Click **`Upload`** to upload the selected file. If Publisher finds any problem with the archive file, view files, or server, it cancels the upload and displays a list of errors. You must correct the problem with the archive file or server before attempting to upload again (you cannot do a partial upload).
7. If Publisher finds no problems, it displays a list of views in the archive. To proceed with the upload, click **`Publish`**. Publisher uploads all version sets in the archive to the current server – *adding to* any existing views and view types already there – and updates relevant database tables accordingly.

Bulk Publishing Examples

One Version Set Published Same Day:

If you publish a dynamic web view version set on 8/7 at 10:00 am, and run the Indexer job at 11:00 am, when a user retrieves the 8/7 statement the next day, the version set published on 8/7 at 10:00 am is used.

Multiple Version Sets Published on the Same Day:

If you publish a version set on 8/7 at 10:00 am, run the Indexer job at 11:00 am, then publish another version set at 3:00, the next day when a user retrieves the 8/7 statement, the 3:00 version set is used, since it was the latest set published on the day the index was created. The time of day the Indexer job ran does not determine which version set is used; the latest set created on that day is used.

Multiple Version Sets Published on Different Days:

A. If a user retrieves a statement with an index date of 10/30, and the associated version sets are dated 10/25, 10/26, 10/31, the 10/26 version set is used.

B. If a user retrieves a statement with an index date of 10/30, and the only associated version sets are dated 11/01 and 11/08, the 11/01 set is used.

Searching for Version Sets

Publisher lets you browse the production environment to list version sets published for an application's dynamic Web views and batch jobs.

You must search for version sets if you want to:

- **Download** individual files or version sets (you cannot download partial version sets).
- **View the content** of individual files.
- **Delete** individual version sets. WARNING: In a normal production environment, you would *never* want to delete a version set. If you accidentally delete a version set, there is *no* recovery plan in place. See "Deleting a Version Set" on page *77* for more information.

You can search for:

- The most recent or all version sets published for an application, a particular dynamic view type and name, or a particular batch job type.
- Version sets published on a specific date for a particular batch job or dynamic Web view.

To search for published version sets (and optionally download or delete):

1. Click **`Manage`** on the Publisher menu. Click the **`Dynamic Web View`** tab (if not already selected) or, to search for a version set for a batch job, click the **`Batch Jobs`** tab.

2. Select an application.

3. For dynamic version sets, you can optionally select view type and view name to narrow your search. For batch jobs, you can optionally choose a particular job type to search on. To view all version sets for the selected criteria, select **`All`** (under Version Sets). To view the most recent version set published for the selected criteria, click **`Latest.`**

   To search for a version set published on a particular day, choose the batch job or a dynamic view type and name, click **`A Specific Day,`** and select the date. Publisher displays all version sets published on that day for the selected criteria.

4. Click **Search**. If the search is successful, Publisher displays the Search Results screen listing each version set found as a timestamp link. If the search is unsuccessful, Publisher displays the message **"There is no version set available to browse."**

5. **To view a list of the files in a version set**, click the version set timestamp link. Publisher displays the Version Set Details screen. You can also download or delete the version set from this screen.

   To view the contents of a design file, click the version set timestamp link to display the file names, and click the particular file link you want to view.

   To delete one or more version sets in the search result, click the check box next to the version set's timestamp link (click the box next to the "Timestamp" heading to select all version sets shown). Click the **Delete Selected** button. Publisher prompts "Are you sure you want to delete?" Verify that you have selected the correct version sets and click OK. You can also delete a single version set from the Version Set Details screen; select the **Delete Set** button on that screen.) WARNING: In a normal production environment, you would *never* want to delete a version set. If you accidentally delete a version set, there is *no* recovery plan in place.

   To download a version set, click the version set timestamp link. Click the **Download Set** button, click **Save**, and specify the archive file name with either a .jar or .zip extension.

To determine how version sets differ:

Although you can see a list of version sets, you may also want to learn how the published version sets differ. Newer version sets could extract new data or have changes to business logic, text, advertisements, etc. You can view the following information about a version set:

- View the timestamp to find the date and time the version set was published.
- Look at the description on the Search Results screen. If Publisher finds a readme.txt for a version set, it displays the first line of the file on the search results screen. Be sure to create a file called readme.txt containing a one-line description of the version set, and publish this file along with the version set. See "Readme.txt Version Set Description" 75 for details.
- Look directly at individual file content. Click the file name in the search results. See "Viewing the Contents of a Design File" on page 76.

Search Results

The Search Results screen shows a timestamp for each published version set that matches your selection criteria.

To view a list of the files in a published version set:

- Click the timestamp link. Publisher displays the Version Set Details screen.

Readme.txt Version Set Description

To help identify and differentiate version sets, you can create and publish a file called readme.txt for each version set.

The readme.txt lets you store 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 file next to each version set in the search results. To view the rest of the contents in a readme.txt, use a text editor or see “Viewing the Contents of a Design File” on page 76.

For consistency, it is a good idea to establish a standard format for the type of information your organization includes in readme.txt files.

You can publish a readme.txt for each version set when publishing individually or in bulk.

To create and publish a readme.txt for a version set:

1. Manually create a file called *readme.txt*.
2. On the first line, enter a one-line description of the version set that you want to appear in Publisher searches. End this line with a paragraph mark (Windows) or with a /R or /N (Unix).
3. You can add additional lines of information to the file, however only the first line appears in Publisher.
4. Publish the readme.txt when you publish the associated version set. When publishing version sets individually, you can browse and select the readme.txt along with the version set files. Include the readme.txt in the view name folder when bulk publishing.

Which Version Set is Used for Live Retrieval

If you’ve published multiple version sets of a dynamic view, the application must determine which associated version set to use for live statement retrieval.

When a user retrieves a statement, edocs determines which published version set to use according to the following rules:

1. **Last version set published on the day the data file was indexed**. The application first looks for and uses a version set published the same day the data file was indexed.

   If more than one version set was published the day the index was created, the set with the latest timestamp before midnight (up to and including 11:59:59) on that day is used. (The time of day the index was created is not saved, only the date, therefore the time Indexer ran is not a factor in determining which version set to use; the version set published latest in the day is always used.)

2. **Most recent version set published before the day the data file was indexed**. If there is no version set published the same day the data file was indexed, the most recent version set published before the date the data file was indexed is used.

3. **Next version set published after the day the data file was indexed.** If there is no version set published on or before the date the data file was indexed, the next version set published after 00:00:00 the next day is used, looking forward for the first available version set.

Maintaining Old Version Sets for Historical Statements

A copy of each version set you publish is maintained; version sets are not replaced or deleted. This lets you maintain a chronological history of application design modifications over time.

Maintaining old version sets also creates a "snapshot" of a statement, freezing the original presentation to enable users to view it unchanged in the future. When a user retrieves an old, or historical statement, the data for that month's statement displays using the version set that was in effect when the data file was indexed.

Warning: Leave old version sets in place to enable the display of historical statements with the original formatting.

For example, if on June 20th, a user retrieves a statement for April 20th, that statement would appear just as it did when originally viewed in April. No data or design changes made to the application after April 20th would apply to the April statement, regardless of when it is viewed. As long as you leave the original version set in place, the original format and content of the April statement remain unchanged.

Because a version set is chosen using the logic described above, it means you cannot make adjustments to a version set published and used with an old statement. For example, if you have version 2.0 of an application in production, and found issues with your version 1.0 application statements, you cannot make adjustments to the version 1.0 version set retroactively. You would only have viewing access to the "snapshot in time" of the version 1.0 statements. If you were to delete the version set published for version 1.0, another version set would be used in accordance with the logic described in *Which Version Set is Used for Live Retrieval*, and inevitably uses a version set not intended for that statement, likely displaying it improperly.

Viewing the Contents of a Design File

You can display the ASCII content of Telco e-Billing Manager ALF, DDF, or XML design files. HTM files display as a static HTML page. You can also view the contents of readme.txt files.

Viewing the contents of a design file lets you determine

- What it is designed to extract or present.

- What modifications were made between different versions of a file (between version sets), such as changes to text, business logic, advertisements, etc.

To view the content of a design file:

1. Search for the version set in Publisher. See “Searching for Version Sets” on page 73 for details.
2. Click the version set timestamp link shown in the search result list.
3. Click the name of the file you want to view. Publisher opens the file and displays the contents.
4. Close the window when finished viewing the file.

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. See *DDF*_Error.log, where DDF is the name of your application, in the temporary folder on the application server for details.

Deleting a Version Set

In a normal production environment, you would *never* want to delete a version set. If you accidentally delete a version set, there is *no* recovery plan in place.

If you delete the only version set associated with an indexed data file, you would no longer be able to view those documents.

If there are several version sets of a view and you delete the latest one, the view published previous to the version set you deleted is used, which can result in the online statement being formatted incorrectly.

Warning: It is recommended that you not delete a version set for any reason, including when archiving old data. The only possible reasons you might want to delete a version set is when you know you made a mistake in publishing it, you are archiving old data (which removes references to the statements in the index) and the version set is associated with that old data, or you are working in a test environment. Be extremely careful if you choose to delete a version set for any reason. Deleting a version set deletes the version set files and any references to the version set in the database.

To delete a version set:

- See “Searching for Version Sets on page 73 for instructions on how to search for and delete one or more version sets from an application.

Version Set Details

The Version Set Details screen shows detailed information about the selected version set, including the names of the files that comprise the version set. It also shows view type, view name, and the version set description from the readme.txt (if any).

To view the contents of a version set file:

- Click the file name link. Publisher displays the file contents.

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. See *DDF*_Error.log, where DDF is the name of your application, in the temporary folder on the application server for details.

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 TBM 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 TBM Enrollment Module; enter the following URL in your browser, substituting your server and application names:
   `http://YourMachineName:PortNumber/eaSample/User?app=UserMain&jsp=/user/jsp/HistoryList.jsp&ddn=ApplicationName`

   (For testing purposes, you can view the NatlWireless application.) TBM displays an enrollment login page.
3. Click the **`Enroll Now`** link. An empty subscription page appears.
4. Enter the customer number. (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**.

4

Managing the Live Production Process

General Production Monitoring Activities

Once you have set up and configured an application and its jobs, you use the edocs Command Center to schedule jobs, manage the production process on a daily basis, and to perform administrative activities related to your application.

The Command Center Main Console provides a high-level status of all activity related to jobs in the production environment, and is the first screen you see when you log into the Command Center.

You also use the Main Console to schedule, control, and monitor all production jobs, including:

- Setting and changing job schedules
- Monitoring the status of jobs and individual production tasks
- Starting a job
- Publishing new version sets
- Monitoring system services

Keeping your applications running efficiently in an ongoing, live production environment requires regular monitoring and maintenance.

Here are a few of the system monitoring activities you want to perform on a regular basis.

Daily application monitoring tasks:

- Check the Command Center Status screen to monitor the state of production jobs.
- Check the administrator email accounts for any administrator alert mail. Administrator email is generated if there's a problem passing email notifications to the SMTP host or if email notification is not working properly for some other reason.

Weekly (or more often) application monitoring tasks:

- Check message log report messages: Activity, Error, and Warning Logs. See “Viewing Message Logs” on page 93.

General system maintenance activities:

- Run activity reports to review application usage statistics.
- Maintain the database. See “Database Administration” on page 99.

Scheduling Jobs

You must manually schedule jobs to run in a live production environment; jobs are not automatically scheduled. The frequency with which you choose to run a job depends on both the job type and your organization's presentment needs. Consider all jobs and system events in planning your schedule.

You can schedule a job to run a simple weekly or monthly basis, or establish a more complex timetable. Review the available scheduling parameters and carefully choose the combination of options that yield the particular schedule you need.

You can change a job schedule anytime, *except while the job is processing*.

Warning: If you try to save schedule changes while a job is running in the production queue (job status says "Processing"), The new scheduling parameters are ignored.

Here is a general idea of how often you might want to run various jobs:

Job Type	How Often to Run
Indexer	Schedule the Indexer job on a regular basis that corresponds with the generation of a new input data file, such as a monthly account statement. You can schedule an Indexer job to rerun frequently over a certain period of time to allow for the expected arrival of a data input file at an unspecific point. (Only when the Scanner task finds a data input file does the job proceed.)
EmailNotification	Schedule EmailNotification to briefly lag the Indexer job so you can verify its completion before notifying users by email that a statement is available.
Purge Apps	Schedule Purge Apps to run as often as necessary to clear space on your database server.
Purge Logs	Schedule Purge Logs to run as often as necessary to clear space on your database server.
HTML Output; XML Output	Schedule an HTML Output or XML Output job on a regular basis that corresponds with the generation of a new input data file, such as a monthly account statement. (HTML or XML output users only). You can schedule HTML Output or XML Output to rerun frequently over a certain period of time to allow for the expected arrival of a data input file at an unspecific point. (The Scanner task runs as part of both job types, and only when it finds a data input file does the job proceed.)
Report	Schedule the Report job to run after the Indexer job (Reporting feature users only).
JIT Report Collector	Schedule the JIT Report Collector when a number of JIT reports have been created, which depends on your system and JIT configuration (Reporting feature users only).

Job Type	How Often to Run
HierarchyImporter and HierarchySyncronizer	Schedule only as needed.
XML Loader	Schedule the XML Loader job on a regular basis that corresponds with the generation of a new input data file, such as a monthly account statement (XML input file users only). You can schedule XML Loader to rerun frequently over a certain period of time to allow for the expected arrival of a data input file at an unspecific point. (Only when the Scanner task finds a data input file does the job proceed.)

The Run Now button

Click the **Run Now** button on the Main Console to run just one instance of the job immediately, overriding the scheduling parameters (except concurrency parameters; if you saved the schedule to run multiple occurrences of a job, the Run Now button uses multiple occurrences instead of one).

Running jobs concurrently (multiple instances)

You can configure the Scheduler to enable multiple instances of an application job to run in parallel. If you do not schedule a job to use concurrency, the job runs sequentially, requiring one job instance to complete before another can start.

If a job processes large or multiple input files, repeating the job sequentially may not allow enough time to complete the job before another input file appeared in the input directory. Running jobs in parallel enables you to leverage machine power to process a large amount of data in less time.

To run instances of a job concurrently, you must configure the maximum number of concurrent job instances to allow for each job (5, 10, 15, or 20) in the job schedule.

Concurrency is available with thread-safe jobs only; PurgeApp and PurgeLogs are not thread-safe and can run only one occurrence at a time.

The Command Center lets you monitor and manage the individual job instances to keep your production environment running efficiently.

To set or change a job schedule:

1. At the edocs Command Center, locate the job you want to schedule or reschedule and click its status in the Next Run column. The Schedule screen appears. (If you just completed configuring a job, the Schedule screen appears automatically.)

2. Specify a valid start date for the schedule to take effect. Click **Popup Calendar** to select dates quickly.

   Specify the schedule window parameters and any repeating parameters, if necessary. These options are described in the following table. To clear the screen to reenter all parameters, click **Clear Schedule**.

Job Schedule Parameters	
Field	**Use to…**
Schedule Date	
Schedule Date	Specify the date the job schedule goes into effect
Schedule Window	
Start Time	Time of day (hour and minutes) to run the job
Try Once	Run the job once on the date and time specified only
Try every … minutes until … time of day	Rerun the job at the specified interval (in minutes) until the end time
Recurring	
Do not repeat this event	Run the job as specified in the previous fields and do not repeat it
Repeat every …	Run the job on the daily or weekly frequency specified: every, every other, every third, or every fourth day, on the selected day of week, all week days, or on both weekend days
Repeat on day … of the month every …	Run the job on the numeric day of the month specified, on every month, every other month, or every 3, 4, 6, or 12 months
Forever	Run the schedule continuously (no end date)
Until …	Run the schedule up until the end date, then stop
Concurrency	
Do not run multiple job instances: only one at a time	Run only one instance of the job at one time.
Run maximum number of (5) concurrent job instances	Run multiple occurrences of the job at one time (concurrently). Specify the maximum number of instances to allow; click the  drop-down box and choose 5, 10, 15, or 20.

3. When finished setting the schedule, click **`Save Schedule`**. This schedule is saved and the job is added to the production queue, overriding any scheduling parameters you set for a single execution of the job. To run the job immediately, click **`Run Now`**. (You can also choose this button on the Main Console. The **`Run Now`** button overrides scheduling parameters except concurrency parameters; if you saved the schedule to run multiple occurrences of a job, the **`Run Now`** button uses multiple occurrences instead of one).

To start a job, list jobs, or view schedules from a command line, see the SDK documentation about implementing the com.edocs.pwc.cli API package.

Monitoring Production Jobs

Use the Command Center Main Console to monitor the state of all production jobs for your applications.

Regularly check the status of jobs and tasks to track:

- Whether a job has completed successfully
- Which tasks completed successfully
- Why a job failed

You can use the Command Center to correct problems, restart or cancel failed jobs, and accept, reject, and purge individual volumes. See “Canceling and Retrying Failed Jobs” on page 90.

For each application, the Main Console lists each configured job type alphabetically. Although there can be multiple instances of an individual job for an application, the Main Console can display only one, so it chooses a representative job instance. The job instances are sorted first by status "ranking" and then by last run time in reverse chronological order. The top-most instance from that list is selected as the representative instance.

Command Center Main Console

Column	Description
Application	Name of the application.
Job Name	Name of the job.
Job Type	The purpose of the batch job: Indexer, Email Notification, Purge Apps, etc.
Last Run	Date and time the representative job instance ran.
Run Time	Elapsed time the representative job instance has been running in hours, minutes, and seconds.
Status	Current execution state of the representative job instance.
Next Run	Date and time the job is scheduled to run next. (This applies only to the job and not a particular instance.)
Action	Displays a button that lets you take action on that job. The **Run Now** button lets you run the job once immediately, overriding the scheduling parameters (except concurrency parameters). The Retry button lets you retry all failed instances of the job.

Note that the Main Console does not show any activity until you create one or more applications and jobs.

To list jobs for a particular application only:

- On the Main Console, click the name of the application in the Application column. The Edit Application page appears, showing only those jobs defined for the selected application.

To sort jobs listed on the Main Console by application:

- Click the word Application in the column header.

To sort jobs on the Main Console by job name (alphabetically), job type, last run, run time, status, or next run:

- Click the column header.

To display the current information on the Main Console:

- Click the **`Refresh`** button.

Viewing Job Status

The status of each production job appears on the Command Center Main Console. Jobs can have the following status, shown here in the order used for ranking purposes:

Job Status	Description
Failed	Job failed
Processing	Job is currently executing
Reprocessing	Job is currently executing after a user manually selected it for reprocessing using the Retry or Retry All button
Reprocess	A user has manually selected the job for reprocessing using the Retry or Retry All button, but the job has not yet begun
No operation	Job/Task did nothing as resources were not ready yet, for example, if the Scanner task found no file in the input directory.
Done	Job has completed successfully
Canceled	Job run failed and was canceled
Not yet started	Job has not begun executing
Done, recurring	Job completed successfully and has been scheduled to run again, or the job has processed one data file and is looking at the input directory to see whether there are any more data files to process in this run
No operation, recurring	Previous job run resulted in a "No operation" status, but the job has been scheduled to run again
Canceled, recurring	Job was canceled and is now looking at the input directory to see whether there are any more data files to process in this run

Viewing and Verifying Task Status

Each production job consists of several individual tasks that work together to generate job output. In addition to job status, each individual task is assigned a status when the job runs. You can closely monitor and manage the task status for a job instance using the Command Center Task Status page.

Every configured task must complete successfully before the application sets job status to Done on the Main Console.

If any of the production tasks is unable to complete, the job fails, and the status changes to Failed. All failed jobs display in red on the Main Console. If a job fails, you can run it again.

To view task status detail for a job:

1. Click the status of the job in the Main Console **Status** column. The Task Status screen appears showing the status of each production task in the job.
2. To change the display order of tasks (processing order remains unchanged), click **Task**. To change the display order of information in the Last Run and Status columns, click **Last Run** or **Status**. Click the links again to restore each display to its original order.
3. Click **Refresh** to display an updated task status.
4. You have the option of rerunning or canceling a failed job. Click **Retry Failed Job**, or **Cancel Failed Job**.
5. The Task Status page displays each instance of a job started during the most recent scheduled run in reverse chronological order (youngest first), along with the status of each task in the instance.

The Task Status page identifies each job instance by a Job Instance ID, and displays the following information:

Command Center Task Status Page

Column	Description
Job Instance ID	A number uniquely identifying each job instance.
Last Updated	The time the task status last updated.
Status	Current execution state of the task. Task Status can be: Processing, Failed, Reprocessing, Reprocess, No operation, Canceled, Not yet started, or Done.
Action	Displays a button that lets you take action on that job instance or on all instances. The Retry button lets you retry that instance; the Retry All button lets you retry all failed instances of the job. The Cancel button lets you cancel that instance; the Cancel All button lets you cancel all failed instances of the job.

Which job instances appear on the Task Status page

The Task Status page displays:

1. Up to the last *N* job instances that have Done, Canceled, or No operation status (where *N* is the maximum number of concurrent instances allowed for the job), *plus*

2. Any instances in Processing, Failed, Reprocessing, or Reprocess status

   If you are not using concurrency (*N*=1), the Task Status page shows up to **five** rows of job instances in Done, Canceled, or No operation status, plus any instances in Processing, Failed, Reprocessing, or Reprocess status.

   When a scheduled run completes, the completed rows remain in view on the Task Status page until a new schedule begins. At this point, the Task Status page begins displaying the instances generated by the new schedule instead. The only exception is that any instances from the previous schedule still in Processing, Failed, Reprocessing, or Reprocess states remain even if a new schedule has begun. Those instances are removed from the Task Status page once processing is complete, or in the case of a failed instance, once you cancel or retry it successfully.

   Schedules can overlap if a second schedule begins before the current run completes. Another scheduled run can begin only if:

3. The first run is not using the maximum number of instances (if enough "resource" is available). For example, if the first run has 3 instances in Processing and the maximum allowed is 10, the next run can start up to 7 new instances.

4. No job instances in the first are in the Failed state.

   Overlapping schedules mean that instances from both schedules could appear on the Task Status page. You can tell from the Last Updated field to which schedule the instance belongs.

   The *number* of rows that appear on the Task Status page at any given time depends on the point of progress of the job plus:

5. Whether you have enabled concurrency for the job (if the maximum number of instances specified in the schedule is >1).

6. The maximum number of concurrent jobs you allow. This number is also the maximum number of Done, Canceled, or No operation jobs that can appear on the Task Status. If you are not using concurrency, the Task Status shows a maximum of 5 job instances in Done, Canceled, or No operation.

7. For jobs that scan for an input file, such as Indexer, the number of input files placed in the input directory.

   For jobs that process multiple statements in parallel with the StatementScanner task, such as the Report job, the number of statements to process up to the maximum number of instances.

8. Whether the job schedule overlaps due to a long lasting run.

Examples: Number of Task Status rows for the Indexer Job

Concurrency Enabled?	Number of Input Files	Maximum Instances	Number of Rows that Can Appear on the Task Status Page
No	3	1	1, 2, or 3. Each job instance must complete before another can appear. If all have been completed, 3 Done instances will show; by default the Task Status page can show up to 5 completed instances when the job runs sequentially (not concurrently).
Yes	3	10	1, 2, or 3. Three job instances can run concurrently and appear at once on the Task Status screen. (Up to 10 rows could appear at once if the job schedules overlap and more input files subsequently appear in the input directory during the second scheduled run.)
Yes	10	3	1-6. As few as one row with status in Processing or, if processing is complete, 3 rows all in Done status can appear. For example, at some point you could see 6 rows; the first instance might be in Processing, the second, third, and fourth in Done, the fifth in Failed, and the sixth in Processing, with the rest not started and therefore not shown.

Additional ways to verify that a task completed successfully

In addition to checking the individual task status on the Task Status screen, you can check for individual task output to determine whether a task completed successfully, as described here:

Task	How to verify task completion
Scanner	Check the job's status window. Verify that the data file was moved from the input directory to the data directory and renamed.
Indexer	Check the job status window. Verify the .IR file is in the */Data/Application Name directory*.
IXLoader Task	Check the job status window. Verify the appearance of .CTL, .CMD, and .LOG files. Check for Index references (one per primary key) within the index table in database.
AutoIndexVolAccept	Check the job status window.
IVNScanner	Check the job status window.
Mail Notification	Check the job status window. Verify emails sent.

Task	How to verify task completion
HTMLFormatter	Check the job status window. Verify the appearance of static HTML output files.
XMLFormatter	Check the job status window. Verify the appearance of static XML output.
StatementsToIR	Check the job status window. Verify the .IR file in the /Data/Application Name directory.
DXLoader	Check the job status window. Verify the existence of a populated database table.

Canceling and Retrying Failed Jobs

You can use the Main Console to cancel or retry a job, and the Task Status page to retry or cancel a failed instance of a job.

If one instance fails, other instances that have started continue to completion, but no new instances are started.

Retry running a failed job or job instance if you want to start it from the point where it failed. If you want to restart a job or instances of a job, cancel and run it again.

If the task has not been started, the Last Update field shows "-" and Status shows Not Yet Started.

To retry a *failed* job before its next scheduled run time:

- On the Main Console, click the **`Retry`** button for the failed job. Or, on the Task Status page, click **`Retry All`**, which retries all failed instances of the job.

  The failed job immediately restarts at the failed task, and changes the instance status from **Failed** to **Reprocess**.

To cancel all instances of a *failed job*:

- On the Task Status page, click the **`Cancel All`** button.

  All failed instances of the job are cancelled, and the job status changes to Canceled, and remains Canceled until the next time the job is scheduled to run again.

To cancel a failed job instance:

- On the Task Status page, click **`Cancel`** in the Action section next to the failed instance.

  The failed job instance is canceled, and the job instance status changes to Canceled, and remains Canceled until the next time the job is scheduled to run again.

Changing a Job Configuration

You change a job configuration any time, *except while the job is processing*.

Warning: If you try to save configuration changes while a job is running in the production queue (job status is "Processing"), the new job configuration parameters are ignored.

To edit a job configuration:

1. On the Main Console, click the name of the job you want to reconfigure. The job configuration screen displays.
2. Enter your changes. If you want to clear all current job parameters, click **`Reset`**.
3. Click Submit Changes and Schedule.
4. Click **`OK`**. The Schedule screen displays, where you can edit the job schedule, if needed.

Deleting a Job

You can delete a job you no longer need in an application. Deleting a job removes the job configuration and schedule in the Command Center.

Deleting a job does not remove data associated with the job that is already in the database. (Use Purge App and Purge Logs jobs to purge data.)

Make sure you really want to delete the job; you can always cancel a job, or change its configuration or schedule.

To delete a job:

1. On the Main Console, click the name of the application. The Edit Application screen displays, which lists the application jobs.
2. Click the box in the "Delete?' column for the job.
3. Click the **`Delete Marked Jobs`** button.
4. Click **`OK`** when asked if you are sure you want to delete the marked job. The job is removed from the application.

Managing Volumes

You can use the Volume Manager to view the status of and manage the availability of volumes processed by the Indexer job for an application (DDN). A volume refers to an individual input file after it has been indexed.

By default, volumes are marked as Accepted if the Indexer job is successful. A volume must be Accepted to be available for viewing on the Web. If errors occur during the Indexer job, the volume is left unmarked.

Use the Volume Manager to:

1. **View volume status** – The Volume Manager page lists each volume processed by the most recent Indexer job for selected application (DDN). A date in either the Date Accepted or Date Rejected column indicates the status of the volume. You can sort the volume list by column headers.

2. **Accept or reject a volume (Intercept to Verify option only**) – If you require manual acceptance of volumes (you're using the Intercept to Verify option in the Action on Index Volume field in the Indexer job configuration), you can use the Volume Manager to manually accept or reject the volume once you have verified the integrity of the file and the associated index data. You manually mark a volume as Accepted to make the content available for Web users (only Accepted volumes are viewable to Web users). You may need to manually reject a volume if you discover an issue with the file or its associated index data. With this feature, a volume shows no status if you have not yet accepted or rejected it. You can also reject a volume after initially accepting it. Once you have rejected a volume, however, you cannot accept it later.

If you must reject a previously accepted volume, you may want to send a new email to clients telling them to disregard the previous notification.

To manage the volumes for an application:

1. On the Command Center Main Console, click the name of the application whose volumes you want to view. The Edit Application screen displays.

2. Click the **`Manage Volumes`** button. The Volume Manager page displays, listing each volume processed by the Indexer job for selected application (DDN). You can sort-by column header or click paging links to display more volumes, if necessary. To display the full pathname of the file, "mouse over" the file name. The IVN (Index Volume Number) column indicates processing sequence.

3. To *accept* a volume, click the **`Accept`** button in the Manage column on the same row as the volume. The volume status changes to Accepted and puts the current date and time in the Date Accepted column, and makes the accepted volume available for viewing on the Web.

   To *reject* a volume, click the **`Reject`** button in the Manage column on the same row as the volume. A message displays warning you that rejecting a volume is irreversible, and asks if you're sure you want to reject the volume. **Caution:** Once you have rejected a volume you cannot accept it later for reactivation. To reject the volume, click **`OK`**. The volume status changes to Rejected, and the current date and time is put in the Date Rejected column, and the rejected volume becomes unavailable for viewing on the Web.

Multiple Statement Processing

Parallel statement processing is used with the **StatementScanner** task in the **Report** job. StatementScanner automatically processes multiple statements from an input file at the same time. An individual Report job instance is created for each statement in the input file it processes.

For example, if you have an input file with 1,000 statements whose accounts are resolved as valid and you set the maximum concurrency for this job at 20, scheduler can start up to 20 job instances to process 20 statements simultaneously.

You can manage the multiple instances of any job that has the StatementScanner task in much the same way you manage concurrent instances generated by other job types. See the information in this chapter about managing job instances using the Main Console and the Task Status page.

Viewing Message Logs

Logs of all activities that occur and messages generated during production are maintained. Review these logs on a regular, ongoing basis to monitor jobs in your production environment.

You can create and view a report showing any of the following types of log messages generated over a select time period:

- **Error** – Error log
- **Information** – Activity log
- **Warning** – Warning log

Log reports display the following information:

Log Report Column	Description
Timestamp	The date and time the message was created in the log
SourceHost	Name of the server that generated the error message or where the production activity occurred
Message ID	A code identifying the task where the error occurred and the level of error
Message	Message text

To view production log messages:

1. Click **`Reporting`** on the Command Center menu. The Reporting screen appears.
2. Click the **`View Logs`** tab to display the View Logs screen.
3. Select the type of message log to view.

4. Enter a start date and end date range to search. Click **`Popup Calendar`** to select dates quickly.
5. Enter a start time and end time to search.
6. Click Submit Query.
7. The log messages display of the selected type generated during the selected date and time range.
8. To select different log information to view, click **`Reselect Log View`**.

Monitoring System Services

You can check on the status of system services using the Command Center.

To view the status of system services:

- Click **`Service Status`** on the Command Center menu. The Service Status page appears, and indicates whether all services are running or which, if any, are missing

If services are missing:

1. Close Command Center.
2. Shut down and restart WebLogic from your WLHome directory.
3. Display the Service Status again to verify that the problem has been corrected. If services are still missing, refer to your Installation Guide.

5 Reviewing System Activity

User Statistics

You can use the Command Center to create and view reports showing detailed information about each time an enrolled user views a statement online. You can search the production database for information about one or all enrolled users, for one or all applications you have, over a particular period of time.

User Statistics reports let you monitor the specific statement viewing activities and habits of individual users.

The following information displays on a User Statistics report:

User Statistic	Description
User ID	The ID of the user who viewed their statement (the name of the user who enrolled)
Date	Date and time the user logged in
Application	Name of the application (DDN) the user viewed
Activity	Type of user action (login, logout, name of view displayed, etc.)

To view user statistics:

1. Click **`Reporting`** on the Command Center menu. The Reporting screen appears.
2. Click the **`User Statistics`** icon. The User Statistics screen appears.
3. Select an application from the drop-down menu.
4. To search for history and statistics on a particular user, enter the user's ID. To see information about all users who have viewed statement during particular time period, leave the User ID field blank.

   Tip: If you don't enter a user ID, the search can take a long time to finish and result in a large data set if you have a large volume of users and traffic. Set a small date and time range to narrow the search.

5. Specify a start and end date range to search. Click **Popup Calendar** to select dates quickly.
6. Specify a start time and end time range to search.
7. Click **Submit Query**. The application's user statistics are displayed for the selected date and time range.
8. Click **Search Again** to perform another search for user statistics.

Statement Statistics

You can use the Command Center to create and view reports showing summary information about how many times users log in and view their statements. You can search the production database for information about statement creation and presentment in one or all applications you may have over a particular time period.

Statement Statistics reports let you monitor the general access volume your applications receive.

You can display a standard ASCII-formatted report or view the statistics graphically in chart format.

The following information displays on a standard Statement Statistics report:

Statistic	Description
# of Logins	The total number of times users logged in during the selected time period
# of Unique Users	The number of users who logged in during the selected time period (does not count repeated logins by the same user)
# of Statements Processed	The number of statements processed during all Indexer tasks run during the selected time period.
# of Statements Viewed	The number of statements viewed during the selected time period (does not count repeated views)

You can optionally display this report as a bar chart.

To view statement statistics:

1. Click **Reporting** on the Command Center menu. The Reporting screen displays.
2. Click the **Bill Statistics** icon to display the Statement Statistics screen. The standard Reports option tab appears by default.
3. If you want the report in the form of a chart, click the **Charts** tab.
4. Select an application from the drop-down menu.

5. For standard reports, enter a start and end date, and a start and end time range to search. Click **Popup Calendar** to select dates quickly. For a chart, select a date range to search from the drop-down list. Your search can be as recent as 24 hours, or as far back as 12 months. Select a report type; you can view number of user logins, unique users, bills processed, or bills viewed.

6. Click **Submit Query** to start the search.

7. The application's statistics report or chart displays for the selected dates.

8. Click **Search Again** to perform another search for statement statistics.

Job Reports

You can use the Command Center to create and view job reports showing history and statistical information about each time a particular job ran in one or more applications over a particular time period. You can generate a report for a specific job or for all jobs.

If you prefer, you can generate a report about jobs that ran against one or more specific data files.

The following information displays on the Job Report for Email Notification:

Column	Description
IVN	Index volume name
Job Name	Name of the job
Application	Name of the application
Start Time	Time the job started
End Time	Time the job ended
Time Elapsed	Total running time
Total Email Count	Total number of emails generated
Total Emails Sent	Total number of emails sent successfully
Total Emails Unsent	Total number of emails not sent; messages can be unsent if the job is still processing or due to an error, exception, etc.
Total Emails Unresolved	Total number of emails with unresolved status, for example, due to missing email address. To calculate this total, you must set the system property "com.edocs.tasks.mns.percentcomplete" to true in the EDX.CONFIG.
Total Emails Failed	Total number of emails with failed status; an email fails if all retries are unsuccessful
Address Error	Number of times an error occurred with the end rolling email address
Server Error	Number of times a mail server error occurred
Job Status	The current status of the job

Column	Description
Data File	Name of the data file the job used
Percent Complete	The percentage of the job that is currently complete. To calculate percent complete, you must set the system property "com.edocs.tasks.mns.percentcomplete" to true in the EDX.CONFIG.

To view a Job Report:

1. Click **`Reporting`** on the Command Center menu. The Reporting screen displays.
2. Click the **`Job Reports`** icon to display the Job Reports screen.
3. Select one or more applications from the drop-down menu.
4. Click Submit Query.
5. To generate a report for a particular job, enter the job name; to get a report for all jobs during a given time frame, leave the job name field blank. **WARNING**: Not entering a job name can return a very large data set that can take a long time to load. For heavily trafficked systems, setting a small date and time range is recommended.
6. Enter a start and end date, and a start and end time range. Click **`Popup Calendar`** to select dates quickly.
7. To generate a report for one or more data files, instead of by job and/or date range, select the files. Tip: To select or unselect a data file, press Ctrl+Left mouse click. WARNING: If you select a data file, all other fields for this query are ignored.
8. Click **`Submit Query`** to start the search.
9. A Job Report for the selected search criteria displays.
10. Click **`Search Again`** to perform another search for job data.

6

Other System Administration Activities

Database Administration

Running an application in a live production environment can generate a large volume of historical data in an application's database. You are responsible for monitoring and maintaining your own database server.

It is a good idea to monitor your database server on a weekly or other regular basis to:

- Check database utilization; to periodically eliminate older application data and free up space on your database server, create, configure, and run the Purge App and Purge Logs jobs.
- Check memory utilization on SQL server/swap (paging) file utilization. When peak number of Commit Charge gets to 10% of limit, then it is advisable to increase the size of paging file, or install more RAM.
- Back up

Purging Historical Application Data; Purge App Job

Create and configure the Purge App job to periodically remove index, email, reporting, detail, annotation, dispute, and PWC data from the database. Running the Purge App job frees up space on your database server.

If your application calls for a certain number of month's statements to be included in a statement summary page, you must regularly run a Purge App job to eliminate index references to old statement data to prevent it from being included in summary page indexed fields.

If you upload detail to the database using the Detail Extractor job, or track annotations and disputes, run Purge App to remove this data as well.

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

To prevent older statements from inclusion in the a summary screen, you must

- Run a Purge Apps job to remove the index references to the data input files that fall outside a specified date range.
- Remove the data input file (if not accessing from primary storage/backup location).

See "**Error! Reference source not found.**" on page **Error! Bookmark not defined.**.

You can also use Purge Logs to clean up a testing or QA machine that has logged data you don't want or need; see "Creating and Configuring a Purge Logs Job" on page 45.

Purging Historical Log Data; Purge Logs Job

Create and configure a Purge Logs job to periodically remove old messages from the log tables in the application database. Over time, log tables can grow significantly in size. Running a Purge Logs job frees up space on your database server.

You can also use Purge Logs to clean up a testing or QA machine that has logged data you don't want or need.

Changing the Administrator's Password

You can change the Administrator's password, which you use to log into the edocs Command Center, at any time.

It's a good idea to periodically change the password to ensure system security.

To change the Administrator's password:

1. On the Command Center Main Console, click **`Settings`**. The Settings screen displays.
2. Click the **`Admin Login`** tab to display the Change Administrator Login page.
3. Enter the new password in the Password field.
4. Enter the password again in the Re-Type Password field to confirm. Click Reset to clear the fields, if necessary.
5. Click Update Password.

Deleting an Old Application

If you have an old or unusable application, you can use the Command Center to delete it from your system.

If you are testing in a quality assurance or development environment, you might want to remove an unneeded application.

You would not normally need to delete an application from a production environment, unless you misconfigured an application or set up the training application by mistake.

Note that deleting an application only removes it from use; it does not delete any associated data in the database for jobs that were run. To delete data from the database, run Purge App and Purge Logs jobs.

To delete an old application:

1. Delete all jobs associated with an application. You can't delete an application until you've deleted all the related jobs.
2. At the Main Console, click the name of the application you want to delete. The Edit Application screen displays.
3. If any jobs still exist, click the box in the "Delete?" column for each job, then click **`Delete Marked Jobs`**. Click **`OK`** when asked if you are sure you want to delete the marked jobs.
4. Click the **`Remove Application`** button (which only appears when no jobs are listed).
5. Click **`OK`** when asked if you are sure you want to delete the application.

7 Configuring Payment Jobs

pmtAllCheckTasks Job

Configuring TBM Payment pmtAllChecksTasks runs the following TBM Payment check payment jobs sequentially.

- pmtRecurringPayment
- pmtCheckSubmit
- pmtCheckUpdate
- pmtPaymentReminder
- pmtSubmitEnroll
- pmtConfirmEnroll
- pmtNotifyEnroll

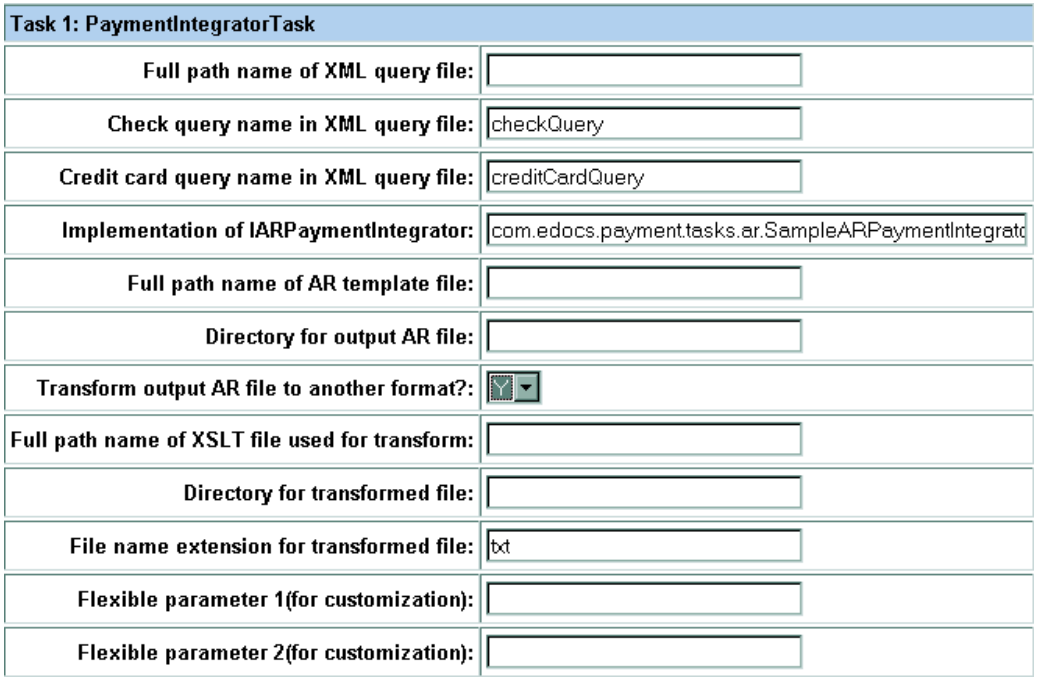
This job presents all the configuration menus from all the other jobs, so you can configure all the listed jobs from this job.

You can also edit pmtAllCheckTasks to not run specific tasks, if you wish to tailor your environment.

pmtARIntegrator Job

pmtARIntegrator uses an XLST template to translate data extracted from payment tables into a different file format. This job runs queries against the *check_payments* and *creditcard_payments* tables to populate a file formatted according to an XML template. Then it uses XLST to transform that data into another file in the format specified by the XLST template.

Configuration



The configurable parameters for this job are:

Full Path of Query Template File - Enter the path to the general query template file. The default *PAYMENT_HOME/lib/payment_resources/ar/ARQuery.xml*, which you can modify to fit your needs. See the example *ARQuery.xml*, where a check query and a credit card query are shown.

Check Query Name in XML Query File- Enter the value of the "name" attribute of the "query" element in *ARQuery.xml*, which is used for the query against the *check_payments* table. This query only works for the *check_payments* and *check_payments_history* tables. You can modify the values, or add new query elements.

Credit Card Query Name in XML Query File - Enter the value of the "name" attribute of the "query" element in *ARQuery.xml*. This query currently only works for the *creditcard_payments* and *creditcard_payments_history* tables You can modify the values, or add new query elements.

Implementation of IARPaymentIntegrator - Specifies the implementation class for IPaymentIntegrator. The default parameter is *com.edocs.tasks.payment.ar.ARPaymentIntegrator*. The IPaymentIntegrator interface defines a method to use TBM Payment to generate A/R files in a specific format. See the *SDK: TBM Payment Module* for information about modifying and implementing a custom class for a plugin.

Full Path of AR Template File - The complete path to the payment source template file. The default parameter is *PAYMENT_HOME/lib/payment_resources/ar/ARFlat_template.txt*. The default file is a sample flat text template file that shows how to format output, which you can edit to meet your requirements. Two other sample template files are provided: *XMLPayment.xml* and *XMLPayment.xml*.

Directory for Output AR File - The directory to put the output file. The default value is *PAYMENT_HOME/Output/ar*.

Transform Output AR File to Another Format? - This flag is ignored unless an XML template is chosen. If it is "Y", the job takes the generated XML file in the output directory as XML input for the XSLT processor, reads the XSLT template, and transforms the data into a different file format.

Do not set this field to "Y" if the XLST file used to transform the AR file to a different format is in TXT format. Only enter Y if the XLST template file format is well formed XML.

Full Pathname of XSLT File Used for Transform - The XSLT template file. You can create your own XSLT template file in a different directory with a different name. The default sample is *PAYMENT_HOME/lib/payment_resources/ar/PaySample.xsl*.

After specifying the preceding parameters and scheduling the job, the A/RIntegrator job will generate an output file in output directory based on your check and credit card query criteria as specified in *ARQuery.xml* and the Java class *ARPaymentIntegrator.java*. You can modify the sample templates files, and re-implement IPaymentIntegrator interface to add features, if desired. For more information about re-implementing the IPaymentIntegrator interface, see the *SDK TBM Payment Module*.

The XLST template file must be well-formed XML, or the pmtARIntegrator job fails with the error:
java.lang.exception: javax.xml.transform.TransformerException.

Directory for Transformed File - The directory for the final transformed file. The default is *PAYMENT_HOME/Output/ar*.

File Name Extension For Transformed File - The file extension of the final transformed file. The default is TXT.

Flexible Parameter 1 (for customization) - Optional parameter that, if provided, will be used in the AR query.

Flexible Parameter 2 (for customization) - Optional parameter that, if provided, will be used in the AR query.

pmtCheckSubmit Job

pmtCheckSubmit selects scheduled check payments that are ready to pay that DDN matches the job's DDN or (optionally) one of the DDNs listed in the Submit Checks for Multiple DDNs field. Checks that are ready to pay are those whose pay dates are scheduled for tomorrow or sooner. It then generates a batch file in the output directory. The output directory is defined in the configuration settings for the payment gateway whose DDN matches the Application.

pmtCheckSubmit uses the check's *pid* to get the latest check account information from the enrollment database, and then uses that to submit the check payment. If the *pid* is null, the check's account information is used for check submission.

A check account may be deactivated, cancelled or physically deleted from the database at the time the scheduled check is submitted. For example, if the check account is deleted, the check will be cancelled instead of submitted. If the check is deactivated to the "pnd_active" or "bad_active" state or is cancelled, you can configure this job to decide whether to cancel the payment or submit it.

A zero dollar amount check (a prenote) won't be submitted, and this job changes the check's status to "processed".

For ACH, you can put checks from other DDNs into the same ACH file, but each DDN must be in a separate batch. The DDNs must have save immediate origination, immediate destination, immediate origination name, and immediate destination name.

The file naming convention for an ACH file is *ppd_yyyyMMddHHmmssSSS.ach*.

The file naming convention for a CheckFree file is *test.ftp_user_name.debit.dat.CCCCMMDDHHMMSS.pgp*. The configuration setting *test* will be replaced by *prod* after the check has gone into production.

The format of the ACH file can be modified by editing the XML files in *<Payment_install_dir>/lib/payment_resources/template/ach* (*<Payment_install_dir>\lib\payment_resources\template\ach* for Windows). See edocs Professional Services for help modifying ACH batch format.

After a check is submitted, its status in the database changes from "scheduled" to "processed". If an error occurs during the check submission process, the status of the check will change to "failed".

The following fields are updated in the *check_payments* table after pmtCheckSubmit runs:

Column updated	Value
last_modify_time	current time
status	7
action_code	For ACH: 27 for checking and 37 for saving. For Checkfree: ADD.
txn_number	For ACH: trace number.

Column updated	Value
reminded	"N"
log_id	id of the summary report in the *payment_log* table.
gateway_payment_id	Populated only if you are using the gateway payment id to match a check returned from ACH to a check in the TBM Payment database. For more information, see the *Customizing and Extending TBM Payment*, or contact edocs Professional Services.
gateway_payment_id	Populated only if you are using gateway payment id to match a check returned from ACH back to a check in the TBM Payment database. For more information, see the *Customizing and Extending TBM Payment*, or contact edocs Professional Services.

Scheduling and Holidays

By default, TBM Payment allows a check payment to be scheduled on a bank holiday. The following rules explain when a Federal holiday qualifies as a bank holiday:

- If the holiday is a weekday, then it is a bank holiday.
- If the holiday is on Sunday, then the following Monday is a bank holiday.
- If the holiday is on Saturday, then even if the previous Friday is a Federal holiday, the holiday is not a bank holiday.

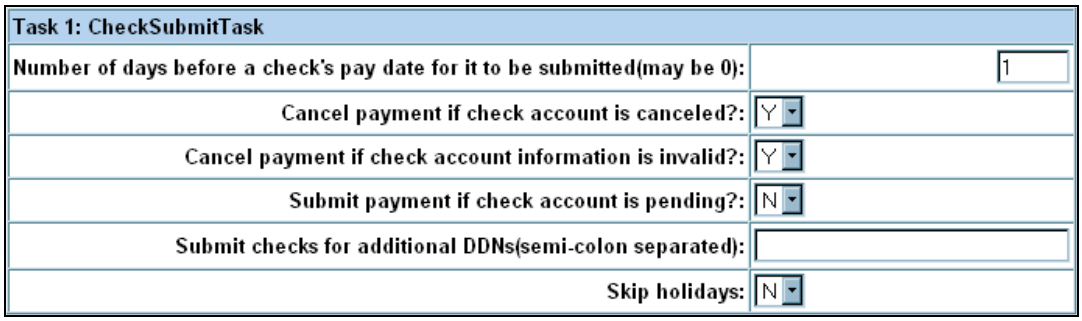
ACH is closed on bank holidays, but it is okay to send a file to ACH which requires transfer of money on holidays: ACH simply processes the checks on the next available bank business day. However, some banks require ACH files to skip bank holidays. By default, TBM Payment skips bank holidays.

When an ACH file is generated, all checks with the same pay dates are put into the same batch, and the batch entry effective date is set. That date is the suggested date for ACH to process the checks in that batch. The following rules determine how the batch entry effective date is set:

- If the pay date is today or earlier, then the batch entry effective date is set to tomorrow . If not, it is set to the pay date.
- If the batch entry effective date is on a bank holiday, then it is moved to the next availed bank business date.

If you don't want to skip holidays when calculating the batch entry effective date, modify the Payment Settings.

Configuration



The configurable parameters for this job are:

Number of days before a check's pay date for it to be submitted - When a check payment is scheduled, a date must be specified when the check is going to be cleared. By default, a check payment will not be submitted to a payment gateway until one day before the scheduled pay date. You can change the submission date by specifying a different value.

Caution Modifying this field may require modifications to the JSP that checks for valid entries when a customer schedules a check.

For example, if the value is one and the job runs today, all checks whose pay dates are tomorrow or earlier will be selected to send to the ACH payment gateway.

Caution Payments made after this job runs will not necessarily be paid on the same day. We recommend running this job at 11:59PM to ensure that payments will be processed on the same day as they were made. If the job runs early in the morning each day (e.g. 2AM), then the job will not process payments scheduled during normal business hours on the same day, since it already ran that day.

Cancel payment if check account is canceled? - Specifies whether the scheduled payment should be canceled if the check account has been cancelled. Normally, this will be "Y". Use "N" if the plugin is going to take actions based on this condition.

Cancel payment if account information is invalid? - The account information (contained in a prenote for ACH or in a payment for CheckFree CDP) sent to the ODFI by the pmtConfirmEnroll job was returned to TBM Payment as having incorrect account information. The user is enrolled, but the account is not valid.

Since the customer's enrollment failed, they will be sent an email when the pmtNotifyEnroll job runs. The customer must resubmit the information for that account, which must be verified before this account can be used to make a payment.

Submit payment if check account is pending? - When the customer adds a new checking account, it is in a pending state until the period specified by Days to Activate Pending Subscribers has expired. "Y" means submit the payment even if the account is pending. "N" means do not submit the payment when the account is pending.

Submit checks for additional DDNs - List any additional DDNs of checks that this job will submit to the payment gateway for processing, separated by semicolons.

Skip Holidays - Determines whether to send the ACH payment batch file to the ACH payment gateway even when the bank is closed because of a holiday. The default is "N", which means send the file even if it is a holiday. The federal holidays are listed on page 124.

pmtCheckUpdate Job

pmtCheckUpdate updates a check's status according to the response from the payment gateway. All files that match the necessary criteria are processed and moved to a history directory under the input directory. The *check_payments* table is examined, and any checks whose status is "processed", and whose pay date is five or more days ago, and has not been returned will have their status changed to "paid".

After the job completes processing, it moves all the processed files to a *history* directory under file input directory.

pmtCheckUpdate Configuration

None required.

ACH Logic

pmtCheckUpdate processes return files from ACH, which can include checks from multiple DDNs. pmtCheckUpdate compares the immediate origin (routing number of the ODFI), immediate destination, immediate origin name, and immediate destination name in the ACH File Header to the same fields that are configured for the ACH payment gateway. If they are the same, pmtCheckUpdate continues processing the ACH return file. If they are not the same, check is ignored.

The order of Immediate Destination and Immediate Origin can be swapped in the ACH return file from what the ACH specification recommends and the pmtCheckUpdate job will still process the file correctly.

For each Batch Header record, pmtCheckUpdate matches the *Company Name* and *Company ID* against the payment gateway definition. If they match, then the current DDN's payment setting is used to process this batch. If either one does not match, pmtCheckUpdate searches the Payment Settings of all DDNs. If it finds a DDN setting with the same immediate information and company information, it uses that setting to process the checks in that batch. If cannot find a match, an exception will be raised and the job will fail.

For each successful Batch Header record match, pmtCheckUpdate updates the status according to the following table:

Addenda Type	Amount field in the first detail record	Return	Status change
99	0	prenote error	prenote_returned
98	0	NOC	noc_returned

Addenda Type	Amount field in the first detail record	Return	Status change
99	not 0	check error	processed

pmtCheckUpdate also updates the status to "paid" if there is no return from ACH after the configured number of days. The number of days to wait after the pay date can be changed in the configuration settings for an ACH payment gateway.

If pmtCheckUpdate will be processing ACH return files containing multiple DDNs, then the Payment Settings for each payment gateway must have the same **immediate origin** and **immediate destination**. Also, each payment gateway used by those DDNs must use the same ACH template files (ACH Template Directory parameter).

ACH Returns

There may be three kinds of returns in one return file:

- check returns
- NOC returns
- prenote returns

For **check returns**, the corresponding check in the *check_payments* table is identified by either payment id or gateway payment id. The check status is updated to "returned", and the **txn_err_msg** field is set to the return code.

The following columns are updated in the *check_payments* table after a check return is processed:

Column updated	Value
last_modify_time	current time
status	-4
reminded	"N"
log_id	id of the exception report in the *payment_log* table
txn_err_msg	ACH return code

For **prenote returns**, the corresponding prenote check in the *check_payments* table is identified by either payment id or gateway payment id. The prenote's status is then updated to "prenote_returned", and the *txn_msg_err* column is set to the return code. From the prenote check, the payer_id (which is the *user_id* column in the *payment_accounts* table) is used to update payment enrollment information. The payment account with the same user id and payment account number will be updated to "bad_active" (*account_status* column) and *txn_message* is set to the ACH return code.

The following columns are updated in the *payment_accounts* table after a prenote is processed:

Column updated	Value
pa#_txn_err	return ACH code
txn_message	return ACH code
pa#_status	bad_active
account_status	bad_active
pa#_txn_notify_status	“N”
notify_status	“N”

NOC Returns

For **NOC returns**, the corresponding check (which can be either regular or prenote) is identified by either payment id or gateway payment id. The NOC's payer_id identifies the corresponding account (Payment matches it with the *user_id* column in the *payment_accounts* table).

If the auto update flag “Update Payment enrollment in Case of NOC” field was set to `true` ("Y") in the Payment Settings, then the corresponding payment account information (routing, acct number or account type) is updated based on the NOC code. If the flag is false ("N"), then the current payment account information is not changed. In either case, the *txn_message* column is populated with the following format:

```
NOC_CODE::NEW_ADDENDA_INFO::OLD_ADDENDA_INFO
NOC_CODE::NEW_ADDENDA_INFO::OLD_ADDENDA_INFO
```

where:

`NOC_CODE` is the returned NOC code,

`NEW_ADDENDA_INFO` is the correct NOC information returned from ACH (content from position 36 to 64 of addenda record, with white spaces trimmed),

`OLD_ADDENDA_INFO` is the existing or incorrect addenda information with the same format as *new_addenda_info* and is calculated on current payment account information.

pa#_notify_status is set to “N” if Notify NOC flag is set to “Y” in Payment Settings. If Send Email Notification in Case of NOC is set to "Y" in Payment Settings, then *notify_status* is set to N". Payment keeps both the old and new payment account addenda information, which allows pmtCheckUpdate to send an email containing both the old and new account information.

The following columns are updated in the *payment_accounts* table after a NOC is processed:

Column updated	Value
pa#_txn_err	NOC_CODE::NEW_ADDENDA_INFO::OLD_ADDENDA_INFO
pa#_number	Changed for C01, C03, C06, C07
txn_message	NOC_CODE::NEW_ADDENDA_INFO::OLD_ADDENDA_INFO
account_number	Changed for C01, C03, C06, C07
pa#_routing	Changed for C02, C03, C07
routing_transit	Changed for C02, C03, C07
pa#_type	Changed for C05, C06, C07
account_type	Changed for C05, C06, C07
pa#_txn_date	current time
txn_date	current time
pa#_txn_notify_status	"N"
notify_status	"N"

ACH Return File Format

The ACH return file should not include new lines at the end of each record. To ensure that the ACH return files are processed correctly, the following information must be returned correctly:

Name	Record	Description
immediateDest	fileHeader	immediate destination must be the same as the one from original ACH file.
immediateOrigin	fileHeader	immediate origination must be the same as the one from original ACH file.
immediateDestName	fileHeader	immediate destination name must be the same as the one from original ACH file.
immediateOriginName	fileHeader	immediate origination name must be the same as the one from original ACH file.
companyName	batchHeader	company name must be the same as the one from original ACH file.
companyId	batchHeader	company id must be the same as the one from original ACH file.
transactionCode	entryDetail	See the ACH specification.
dfiAcctNumber	entryDetail	Check account number; see the ACH specification.
amount	entryDetail	Amount of original check.
individualId	entryDetail	Must be the same as the one from original ACH file.

Name	Record	Description
individualName	entryDetail	Must be the same as the one from original ACH file.
addendaTypeCode	addenda	98: NOC return; 99: check return. See the ACH specification for details
returnReasonCode	addenda	return code.
originalEntryTraceNumber	addenda	The trace number of the check from the original file sent to ACH.
originalRDFI	addenda	Copy data from positions 04-11 of the original Entry Detail Record. See the ACH specification.
entryAddendaCount	fileTrailer	Total number of addenda returned in the file.
totalDebitAmount	fileTrailer	Total debit amount in the file.
totalCreditAmount	fileTrailer	Total credit amount in the file.

CheckFree Logic

pmtCheckUpdate only processes the files belonging to the payee or DDN of this job.

pmtCheckUpdate checks the third field of the file to determine if the file type is confirm, trnsumm, trnjrnl, setlmnt, or returns. It then verifies that the ClientID field in the file matches the Client ID setting for this gateway configuration. pmtCheckUpdate updates the status of the checks in each file as follows:

File Type	Action
confirm, trnjrnl	Changes the status from processed to paid for all the checks whose pay date is five days before the current day and have no error. An error message results in the status of the checks in that file being changed to failed.
returns	Status of checks in this file will be updated to returned.
transumm, setlmnt	These files are not processed, but are still moved to the history directory.

pmtConfirmEnroll Job

This job only applies to the ACH payment gateway.

The pmtConfirmEnroll job checks the *txn_date* field in the *payment_accounts* table to find each new account (the *account_status* field is "pnd_wait") If the specified number of days have passed since the user signed up for enrollment (*txn_date*), pmtConfirmEnroll updates the *account_status* field to "active".

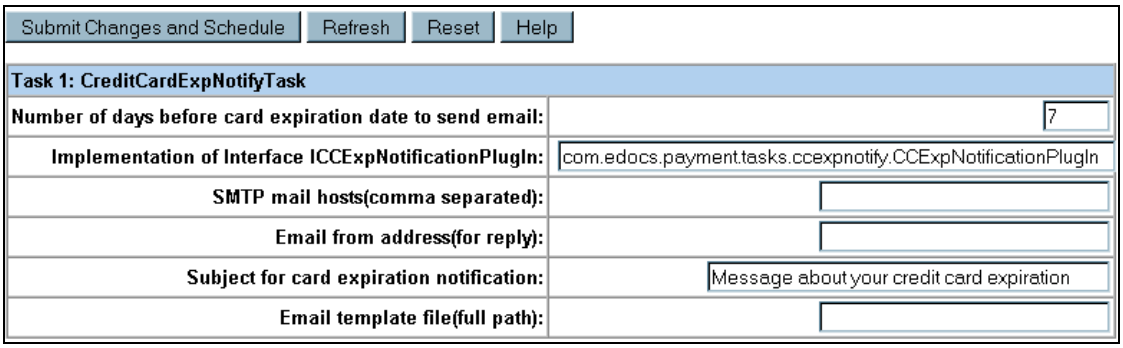
The number of days to wait is specified by the Days To Activate Pending Subscribers parameter in the Payment Settings for the ACH payment gateway.

pmtConfirmEnroll Configuration

No configuration is required when creating this job.

pmtCreditCardExpNotify

For users that have configured a credit card account for payments, this job sends an email to users whose credit card will expire soon.



The configurable parameters for this job are:

Number of days before card expiration date to send email - Specifies the number of days before their credit card expires to send the expiration notice. The default is 7.

Implementation of Interface ICCExpNotificationPlugin - This is the name of the java class that is called before the pmtCreditCardExpNotify job emails notifications. It currently does nothing, but you can replace this class with one of your own to process additional business logic to and possibly modify credit card expiration email, or cancel it completely.

For information about implementing this class, contact edocs Professional Services. The default value is: *com.edocs.payment.tasks.ccexpnotify.CCExpNotificationPlugIn*, which takes no action.

SMTP mail hosts - Name(s) of mail host system.

Email from address - Sender's email address.

Subject for card expiration notice - Text message to appear on email subject line.

Email template file - Full directory path to the formatted message template file.

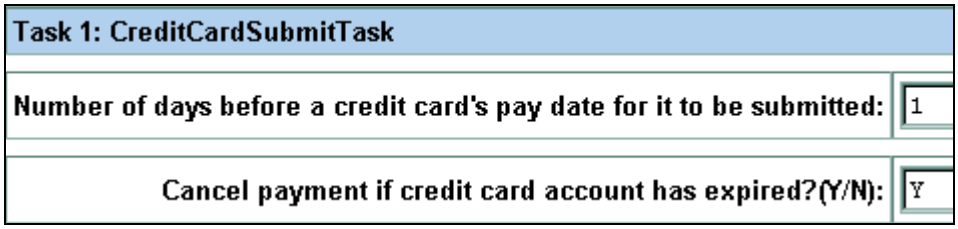
For Windows, this path is
%PAYMENT_HOME%\lib\payment_resources\CCExpNotify.txt

For Unix, the path is *$PAYMENT_HOME/lib/payment_resources/CCExpNotify.txt*

pmtCreditCardSubmit Job

This job selects credit card payments that are scheduled to be paid within a configurable number of days before today, and opens a connection to a credit card payment gateway to authorize and settle those transactions. Both authorization and payment are done at the same time.

pmtCreditCardSubmit Configuration



The configurable parameters for this job are:

Number of Days Before a Credit Card's Pay Date for it to be Submitted - When a credit card payment is scheduled, a date must be specified when the credit card payment should be settled. By default, a credit card payment will not be submitted to a payment gateway until one day before the scheduled pay date. The submission date can be changed by specifying a different value.

Caution Modifying this field may require modifications to the JSP that checks for valid entries when a customer schedules a payment.

Cancel Payment if Credit Card Account has Expired - "Y" means if the credit card account used in a payment has expired, then cancel the payment. "N" means try to make the payment with the old account, failure of which causes email to be sent to the customer, if configured by the pmtPaymentReminder job.

pmtCreditCardSubmit Operation

pmtCreditCardSubmit submits credit cards to a credit card gateway to be processed. It searches the *creditcard_payments* table to find all scheduled credit card payments whose *status* field is "scheduled" (6) and whose *pay_date* field has a date the same as or prior to one day after the day the job is running (by default), and sends them the credit card gateway for processing.

Credit card account information is saved by TBM Payment as part of the payment when the payment is scheduled. Whether this copy of the account information is used for submission depends on the contents of the *pid* field.

- When *pid* is **not null**, the saved account information is used to extract the latest credit card account information from the enrollment database, and the extracted account information is used for submission. This eliminates any potential problems related to changing or deleting a credit card account after the payment is scheduled. The Cancel Payment if Credit Card Account has Expired parameter determines which action to take with scheduled payments when the credit card account is changed.

- When the *pid* is **null**, the saved copy of the credit card account is used. This is useful when the pid cannot be found in enrollment database. For example, when a customer database offers payment account information but doesn't have a unique *pid*.

If pmtCreditCardSubmit is successful submitting the credit card payment, the payment is approved, money is guaranteed to be transferred, and the status of the payment is set to "settled" (8). If there is a problem submitting the payment, its status is set to "failed-authorize" (-4).

TBM Payment supports the Verisign credit card cartridge and uses HTTP to communicate with it. If there is a network problem, the *status* of the payment stays "scheduled", but the payment's *txn_err_msg* field gets the error message. This ensures that the payment will be picked up by the next run of the pmtCreditCardSubmit job. If the payment is successful, Payment stores the confirmation number from Verisign in the *txn_number* field of the *creditcard_payments* table.

Verisign Threads

To speed up credit card processing, you can configure the Verisign cartridge to use simultaneous connections (threads) with Verisign. By default, the Number of Threads field in the Payment Settings is 1, but you can enter a larger number to speed processing.

However, there is a bug with the Verisign SDK, which causes a connection failure when the number of threads is too high. This problem is especially noticeable NT, but also occurs on UNIX. On NT, the number of threads should not exceed 10. Connection failures can be significantly reduced by using multiple copies of the Verisign certificates. By default, there is only one certificate.

Connection failures caused by the Verisign bug are not fatal. Payment recognizes the failure and keeps the payment's status as scheduled, so that the failed payments will be processed the next time pmtCreditCardSubmit runs. If you increase the number of threads and find there are failures, schedule your job run twice, back to back.

The following columns are updated after a credit card is submitted:

Column updated	Value
last_modify_time	Current submit time.
reminded	Set to "N" if the status is "settled" or "failed-authorize".
status	Set to "settled" if the payment is accepted by card issuer. Set to "failed-authorize" if the payment is rejected or returned as "referral" by card issuer. Stays "scheduled" if there is a network error.
log_id	Contains the value in *payment_log*, which represents a report id.

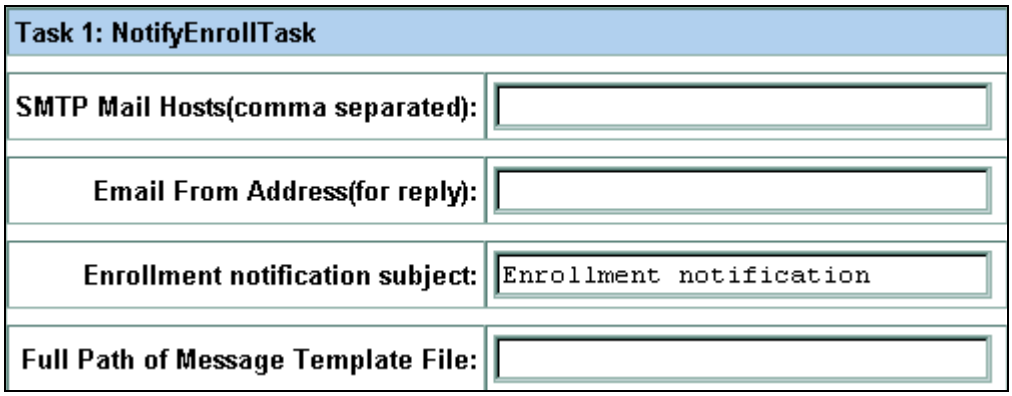
pmtCustom

The pmtCustom job can be customized to perform tasks that are not part of other TBM Payment jobs. See the *Customizing and Extending TBM Payment* document for information about using the pmtCustom job.

pmtNotifyEnroll Job

Sends email notification to customers about the status of their payment account activation, plus account information changes as a result of ACH NOC returns .

Creating a new pmtNotifyEnroll job in the Command Center of type pmtNotifyEnroll returns a dialog similar to the following:



The configurable parameters for pmtNotifyEnroll are:

SMTP Mail Hosts -Name(s) of mail host system.

Email From Address - Sender's email address.

Enrollment notification subject - Text message to appear on email subject line.

Full Path of Message Template File - Full directory path to the formatted message template file.

For Windows, this path is *%PAYMENT_HOME%\lib\payment_resources\NotifyEnroll.txt*.

For Unix, the path is *$PAYMENT_HOME/lib/payment_resources/NotifyEnroll.txt*.

pmtNotifyEnroll Job Email Format

pmtNotifyEnroll sends out email notifications to customers who have successfully enrolled, and to customers who have had problems with their enrollment.

For ACH, pmtNotifyEnroll finds payment accounts in the *payment_accounts* table whose *notify_status* field is "N". It then checks the *account_status* field, and if it is "active" or "bad_active", it sends an email to the customer about the status of their account. After sending email, pmtNotifyEnroll changes the *notify_status* field to "Y". pmtNotifyEnroll does not send emails if *account_status* is "pnd_active" even though the *notify_status* is "N".

For Checkfree, pmtNotifyEnroll finds payment accounts in the *payment_accounts* table whose *notify_status* is "N", sends email, and changes the *notify_status* to "Y".

The format of the email message that is sent is generated based on the contents of an email template text file. The Full Path of Message Template File configuration parameter specifies the location of this template file. The default is *<install_dir>/lib/payment_resources/NotifyEnroll.txt*.

pmtPaymentReminder Job

The pmtPaymentReminder job sends payment email notifications to customers:

- Who have configured payment reminders
- When a check's status changes to processed, failed, canceled, or returned.

pmtPaymentReminder Job Configuration

Task 1: PaymentReminderTask	
SMTP mail hosts(comma separated):	
Email from address(for reply):	
Email template file(full path):	
Implementation of interface IPaymentReminderPlugIn:	com.edocs.payment.tasks.reminder.PaymentReminderPlug
Subject for payment reminder:	Please pay your bill
Notify if a check is sent for processing?:	N
Subject for processed check:	Your check has been sent for processing
Notify if a check is cleared?:	Y
Subject for cleared check:	Your check has been cleared
Notify if a check is returned or failed or canceled by eaPay?:	Y
Subject for returned/failed/canceled check:	There is a problem to process your check
Notify if a credit card is settled?:	Y
Subject for settled credit card:	Your credit card has been settled
Notify if a credit card is failed-authorize or canceled by eaPay?:	Y
Subject for failed-authorize/canceled credit card:	There is a problem to process your credit card

The configurable parameters for this job are:

SMTP Mail Hosts - Name of the mail host.

The current version of TBM Payment **does not** support user authentication for SMTP servers. If the SMTP server requires a user name and password for sending mail, an exception must be made for email originating from the machine on which the payment Application Server is installed.

Email From Address (for reply) - Sender's email address, which the customer's email will use as Reply-To.

Email Template File (full path)- Full directory path to the formatted message template file. This template file is used to generate the body of email messages. It can be customized.

The default path to the formatted message template file for Windows is *%PAYMENT_HOME%\lib\payment_resources\paymentreminder.txt*.

For Unix it is *$PAYMENT_HOME/payment/lib/payment_resources/paymentreminder.txt*.

Implementation of Interface IPaymentReminderPlugin - The plugin allows modification of whether a payment reminder is sent, plus other actions based on the type of reminder. For information about implementing this class, contact edocs Professional Services. The default is *com.edocs.payment.tasks.reminder.PaymentReminderPlugin*.

Subject for payment reminder - Enter the text to use for the subject line of emails sent to users to remind them that a payment is about to become due.

Notify if a check is sent for processing - Indicates whether notification is to be sent for checks that were sent for processing.

Subject for processed check - Enter the text to use for the subject line of emails sent to users to notify them that a check has been sent for processing (to make a payment).

Notify if a check is cleared - Indicates whether notification is to be sent for checks that have been paid (cleared).

Subject for cleared check - Enter the text to use for the subject line of emails sent to users to notify them that a check has cleared.

Notify if a check is returned or failed or canceled by TBM Payment- Indicates whether notification is to be sent for checks that were returned by the payment gateway as canceled, returned or failed settlement.

Subject for returned/failed/canceled check - Enter the text to use for the subject line of emails sent to users to notify them that a check has did not settle because it was returned, failed or was canceled. The ACH return file will explain the exact cause.

Notify if a credit card is settled - Indicates whether email should be sent for credit card payments that settled successfully.

Subject for settled credit card - Enter the text to use for the subject line of emails sent to users to notify them that a credit card payment has settled.

Notify if a credit card is failed-authorize or canceled by TBM Payment - Indicates whether email should be sent for credit card payments that failed to authorize or were canceled by TBM Payment.

Subject for failed-authorize/canceled credit card - Enter the text to use for the subject line of emails sent to users to notify them that a credit card payment has either failed to authorize or the card has been cancelled.

pmtPaymentReminder Operation

Payment Reminders

Email notifications are sent out for customers that have set up payment reminders. pmtPaymentReminder sends out reminder email for reminders in the *payment_reminders* table whose *next_reminder_date* is today or later, and whose *Reminded* field is "N". After sending the email, the *Reminded* field is updated to "Y", and the *next_reminder_date* field is updated as specified by the *reminder_interval* field.

Check Payment Notification

pmtPaymentReminder sends out email for checks as configured in the job. Email notifications can be sent for rows in the *check_payments* table where the *status* field is "Returned", "Failed", or "Processed" (sent for processing). After sending the email, the *Reminded* field is updated to "Y".

The valid values for the *Status* field are:

Status	Value
Returned	-4
Failed	-1
Scheduled	6
Processed	7
Paid	8
Canceled	9

Credit Card Payments

pmtPaymentReminder sends email for the scheduled credit cards in the *creditcard_payments* table whose *status* field is "settled" or "failed to authorize", and whose *reminded* field is "N". After sending email, pmtPaymentReminder sets the *reminded* field to "Y". pmtCreditCardSubmit sets the reminded field back to "N" when it makes a payment for that scheduled credit card.

Normally, no emails are sent for instant credit card payments. You can enable email notification for instant credit card payments by editing the *instantPayment.jsp* file to set the *reminded* field to "N". The instantPayment.jsp file is in web\payment\user\jsp when you un-jar the EAR file.

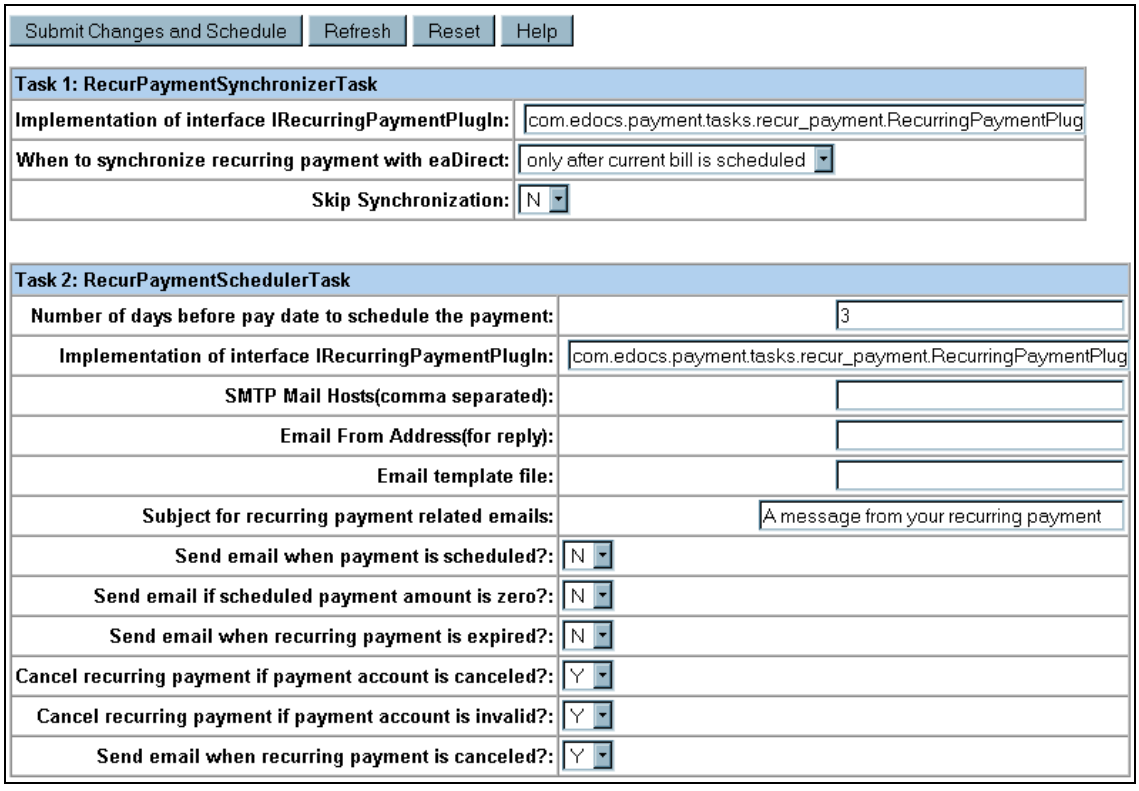
Email Template

The email address is retrieved from the *payer_email_addr* field in the *payment_reminders* table. Email content is created using the email template file configured for this job. The default template file is *paymentReminder.txt*, which is in the *<install_dir>/lib/payment_resources* directory. See the section ***Error! Reference source not found.*** for information about editing the email template to modify email content.

pmtRecurringPayment Job

The pmtRecurringPayment job checks for recurring payments that are due for payment and schedules them to be paid. It also optionally sends email to the customer when it schedules a payment, so that the customer can modify or cancel the scheduled payment before the payment is made.

pmtRecurringPayment Configuration



The configurable parameters for this job are:

Task 1: RecurPaymentSynchronizerTask

Implementation of interface IRecurringPaymentPlugIn - This is the name of the java class that is called before the pmtRecurringPayment job schedules a payment. It currently does nothing, but you can replace this class with one of your own to process additional business logic to and possibly modify how the payment is scheduled, or cancel it completely. For example, to copy selected fields from an index table into a payment table, or to deny a recurring payment.

For information about implementing this class, contact edocs Professional Services. The default value is: *com.edocs.payment.tasks.recur_payment.IRecurringPaymentPlugIn*, which takes no action.

When to synchronize recurring payment with TBM - By default, TBM Payment uses the latest available bill when submitting the payment to the payment gateway. You can configure each payment gateway to only synchronize once, which reduces processing. The setting "whenever job runs" can be changed to "Only after the current bill is scheduled", which causes Payment to synchronize only once; when the bill is scheduled.

Skip Synchronization: N (default) enables synchronization. If you wish to ignore synchronization and start scheduling immediately, then change this to **Y**.

Task 2: RecurPaymentSchedulerTask

Implementation of interface IRecurringPaymentPlugIn - This is the name of the java class that is called before the pmtRecurringPayment job schedules a payment. It currently does nothing, but you can replace this class with one of your own to process additional business logic to and possibly modify how the payment is scheduled, or cancel it completely. For example, to copy selected fields from an index table into a payment table, or to deny a recurring payment.

For information about implementing this class, contact edocs Professional Services. The default value is: *com.edocs.payment.tasks.recur_payment.IRecurringPaymentPlugIn*, which takes no action.

Number of days before pay date to schedule the payment – The check payment will be scheduled *N* days before the pay date by pmtRecurringPayment. *N* days before the due date, email notification will be sent, if Send email notification when the payment is scheduled is set to "Y". That gives the customer *N* days (less one, the day it is paid) to modify or cancel the scheduled payment.

Modifying this field may require modifications to the JSP that checks for valid entries when a customer schedules a check.

SMTP Mail Hosts - List all the SMTP hosts that are available to send email notifications. This is not required if *Send email notification when the payment is scheduled* is **N**.

Email From Address (for Reply) - The email address that replies should be sent to.

Email template file - The email template file to use when creating a recurring payment notification email.

Subject for recurring payment related emails - Enter the text to use for the subject line of emails sent to users to notify them that a recurring payment has scheduled a payment.

Send email when payment is scheduled - Determines whether email notification is active for recurring payments.

Send email when recurring payment is expired - Determines whether email notification is sent when a recurring payment effective period has ended.

Cancel recurring payment if payment account is canceled? - Specify whether the recurring payment should be canceled if the account has been cancelled. This will be considered recurring payment expiration, so an email will be sent to the user.

Normally, this parameter will be "Y". Use "N" to have the pmtCheckSubmit job handle this condition, or if the plugin is going to take actions based on this condition.

Cancel recurring payment if payment account information is invalid? - The account information (contained in a prenote for ACH or in a payment for CheckFree CDP) sent to the ODFI by the pmtConfirmEnroll job was returned to TBM Payment as having incorrect account information. The user is enrolled, but the account is not valid.

If a credit card account was used for enrollment, the account information is not checked until a payment is made. If a credit card payment is sent to the payment processor with invalid account information, the account will be marked invalid.

Since the customer's enrollment failed, they will be sent an email when the pmtNotifyEnroll job runs. The customer must resubmit the information for that account, which must be verified before this account can be used to make a payment.

Send email when payment account is canceled? - Specify whether to send email to notify the user that their payment account was cancelled.

pmtRecurringPayment Operation

pmtRecurringPayment examines all the customer accounts to see which recurring payments need to be scheduled. It looks for recurring payments where the amount due is greater than zero, and the date to be scheduled is equal to or greater than *n* days before the current date, where *n* is configured on Number of days before pay date to schedule the payment .

If the number of payments specified in the **effective period** on the customer interface has been met, this job sets the recurring payment to "inactive".

If the customer selects a payment to be made based on the number of days before the due date, or selects the amount to be based on the due amount, pmtRecurringPayment must query TBM to determine when to schedule the payment and/or how much to pay. For that reason, the pmtRecurringPayment job should be run after the TBM Indexer job runs.

If the customer selects a payment for a fixed amount on a fixed date, then the pmtRecurringPayment job does not need to query TBM to schedule the payment.

For additional information about how recurring payments work and the different types, see the section Recurring Payments.

pmtSubmitEnroll Job

ACH optionally accepts enrollment information to verify the customer's check routing number and check account number. ACH calls this enrollment information a **prenote** which is the same as a regular check payment, except its dollar amount is zero. The name of the generated ACH file is *ppd_yyyyMMddHHmmssSSS.ach*.

pmtSubmitEnroll submits enrollment information to a payment gateway (for ACH only). It finds all accounts in the *payment_accounts* table whose *account_status* field is "pnd_active" and writes them into an ACH file. The *txn_date* field is set to the current date, and *account_status* field is changed to "pnd_wait".

pmtSubmitEnroll Configuration

The configurable parameter for this job are:

Skip Holidays - Determines whether to send the ACH payment batch file to the ACH payment gateway even when the bank is closed because of a holiday. The default is "N", which means send the file even if it is a holiday.

ACH Federal Holidays

ACH check payment gateways have the field *Skip non-business days for batch effective entry date* to determine when a payment should be made. Non-business days in TBM Payment include the following U.S. federal holidays:

Holiday	Date
New Years Day	January 1
Martin Luther King's Birthday	Third Monday in January
Presidents' Day	Third Monday in February
Memorial Day	Last Monday in May
Independence Day	July 4
Labor Day	First Monday in September
Columbus Day	Second Monday in October

Holiday	Date
Veterans' Day	November 11
Thanksgiving	Fourth Thursday in November
Christmas Day	December 25

If a U.S. federal holiday falls on a Saturday, then the previous Friday is a holiday for Federal employees, but it is not a holiday for most businesses and employees.

8 Payment Reports

Overview

TBM Payment keeps payment history for auditing purposes. Checks go through a list of states before it is cleared. For insert, update/insert and update operations, Payment keeps a copy of the in the *check_payments_history* table. TBM Payment records when the check was created, when the check was updated or cancelled, when the check is processed, and other check actions.

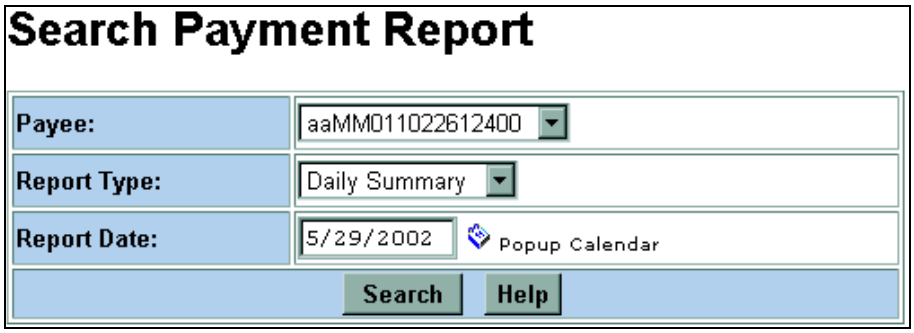
TBM Payment also logs additional important information: warnings and errors. Based on the format of the logging messages, Payment TBM Payment can hook up with other monitor systems, such as EMC patrol.

Viewing TBM Payment Reports

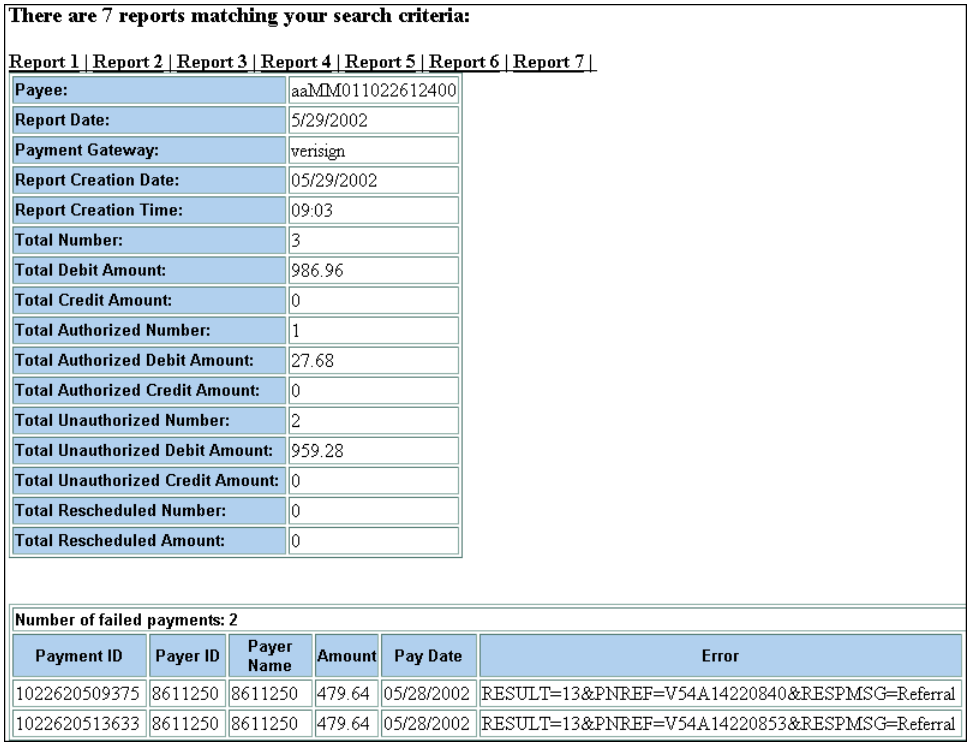
After a payment job runs, payment reports can be viewed for entries from that job based on two report types: Daily Summary and Daily Exceptions.

To view TBM Payment reports:

1. In the Command Center, click **`Reporting`**. The Reporting and Log page appears.
2. Select the **`Payment Reports`** tab. The Search Payment Report page appears.
3. Select a Payee from the drop-down menu and specify a Report Type.
4. Enter a date on which you want to perform the search. You can use the Popup Calendar to help you determine the date.



5. Click **Search**. Depending on the type of payment report specified in the Report Type field (Daily Summary or Daily Exception), you will see reports similar to the following:



There are 7 reports matching your search criteria:

Report 1 | Report 2 | Report 3 | Report 4 | Report 5 | Report 6 | Report 7 |

Payee:	aaMM011022612400
Report Date:	5/29/2002
Payment Gateway:	verisign
Report Creation Date:	05/29/2002
Report Creation Time:	09:03
Total Number:	3
Total Debit Amount:	986.96
Total Credit Amount:	0
Total Authorized Number:	1
Total Authorized Debit Amount:	27.68
Total Authorized Credit Amount:	0
Total Unauthorized Number:	2
Total Unauthorized Debit Amount:	959.28
Total Unauthorized Credit Amount:	0
Total Rescheduled Number:	0
Total Rescheduled Amount:	0

Number of failed payments: 2

Payment ID	Payer ID	Payer Name	Amount	Pay Date	Error
1022620509375	8611250	8611250	479.64	05/28/2002	RESULT=13&PNREF=V54A14220840&RESPMSG=Referral
1022620513633	8611250	8611250	479.64	05/28/2002	RESULT=13&PNREF=V54A14220853&RESPMSG=Referral

Credit Card Gateways

Credit card gateways will only report credit card transactions.

Viewing TBM Payment Module Error Logs

TBM Payment jobs may generate error messages. TBM Payment writes all errors to the payment database table, and appends them with the *.log* suffix. Error logs can be viewed through the Reporting menu option in the Command Center.

The following table lists TBM Payment Module error messages.

Error Message	Description
NoPaymentAccountException	Indicates an ACH customer cannot be activated (the ACH payment gateway only supports up to three payment accounts for each customer) and can result in the failure of the payment application. This type of error typically indicates a problem with the Enrollment JSP page and should be discovered during initial system setup and testing.
FileIOException	Indicates there is a problem regarding the reading and writing of ACH files. Verify the existence of the ACH file output directory and file input directory, and make sure the permissions on them are correct.
CassetteException	Indicates there is a general problem regarding an ACH operation and the interpretation of the error depends on the specific error messages themselves. This error message is sometimes used as a wrapper for other errors.
AchFormatException	Indicates there is a problem with the ACH file format. For example, one common source of this error is that the length of supplied data is greater than the length allowed by ACH.
TemplateException	Indicates one or more ACH template files failed during processing by the Template engine. In some cases, this error can be corrected during system setup and configuration. Also check the ACH template formats to make sure they are valid.
SQLException	Indicates a generic exception accessing the payment database. This error message can be generated by a variety of errors, but one common source is the failure to establish connection pools in the payment database. In this case, check whether the database server is still running, and check the application server configuration.
OperationNotSupported Exception	Indicates the current payment operation is not supported by ACH. This error message typically indicates faulty coding and should be corrected by a developer immediately.
RemoteException	Indicates a generic exception and typically serves as the wrapper for other exceptions. This error message typically indicates a failed payment operation, which cannot be corrected by running the operation again.

Error Message	Description
DuplicateKeyException	Indicates an exception that occurs when there are two check payments in the database with the same payment id. TBM Payment uses a timestamp (in milliseconds) to assign payment id's. This type of error typically does not require manual intervention. Instead, the customer will receive an error message and then be prompted to submit the check payment again.
NoDataFoundException	Indicates there is no data being extracted from the payment database. This type of error is typically corrected by the application during processing.

9

Payment Administration

Payment Database

Preventing Multiple Payments

By default, TBM Payment allows a bill to be paid more than once. To ensure that a bill can only be paid once, you need to add a unique key constraint on the *bill_id* field of the *check_payments* table. Run *$PAYMENT_HOME/db/set_unique_bill_id.sql* to set the unique constraint. The *bill_id* in Pay is the same as the doc id in TBM.

If a customer tries to pay a bill that has already been paid after the unique key constraint has been added (either from the UI or by a previously scheduled recurring payment), the customer will receive an error message stating that the bill has been already paid. If the bill is paid from the UI and a recurring payment tries to pay it again, the payment will fail and an email notification message will be sent to the customer (if recurring payments are configured for that email notification).

Adding this constraint won't prevent a customer from making a payment using a bill id. For example, a customer can still make a payment directly from the **Make Check Payment** link, which allows them to make a payment without specifying a bill.

The unique key constraint only informs a customer that the bill has been paid when they try to pay a bill that has already been paid. If you want to provide additional features, such as disabling the payment button when the bill has already been paid, you must query the database to get that information. Use caution when adding extra functions because performing additional database queries can deteriorate TBM Payment performance. Make sure to create the proper index if you plan to create a new query.

UI Actions and Database Changes

The following table lists user actions available in the example interface and the edocs Command Center Payment jobs, and describes their impact on the TBM/Payment database. This example uses the enrollment for ACH where Payment keeps payment information in a separate database:

UI Action	Payment Database
Create payment setting (Command Center)	Payment setting information is saved in the *payment_profile* table. The param_name *user_account_accessor* should point to the right *IUserAccountAccessor* implementation and *payment_account_accessor* should point to the right *IPaymentAccountAccessor* implementation.
User enrolls	User information is inserted into CDA and payment accounts are inserted into *payment_accounts*.
Run pmtSubmitEnroll	Finds all payment accounts whose *account_status* is "pnd_active", *txn_date* is "null", and sends to the ACH payment gateway. After that, it sets *pa#_txn_date* to the current date (yyyyMMddHHmm).
Run pmtConfirmEnroll	Changes *pa#_txn_status* to "bad_active" if returned, or to "active" if there is no return after three days. Updates *pa#_txn_notify_status* to "N".
Run pmtNotifyEnroll	Finds all payment accounts whose *pa#_txn_notify_status* is "N", and send emails. Updates *pa#_txn_notify_status* to "Y".
User logs in	The user's id and password is checked against the *user_id* and hash in the enrollment table.
User makes a check payment	The payment is saved into the *check_payments* table. The status of the check is "scheduled"(6).
User clicks on **`Future Payments`**	Displays a list of scheduled payments for this user. The user can cancel or update scheduled payments from here.
User cancels or updates a check	When a user cancels a check, the status of that check is set to "canceled" (9). The check is not deleted from the database. When a user updates a check, the same entry in *check_payments* will be updated. A check can only be cancelled or updated when the status of that check is "scheduled"(6).
Run pmtCheckSubmit	Finds all checks due by tomorrow (or before), and sends them to the ACH payment gateway. The status of the check changed to "processed"(7). *txn_number* now holds the trace number of the check, and the *reminded* field is set to "N". A batch report file is written to the *payment_log* table, whose type is *summary*. You can view this report from the Command Center.
User clicks on **`Payment History`**	Processed check payments will show here. Paid, returned, failed and cancelled checks will also show here.

UI Action	Payment Database
Run pmtCheckUpdate	Changes the check status to "returned" (-4) if there is a return for it, or to "paid" if there is no return after five business days and *reminded* field is set to **N**. If the check is returned, then the *txn_error_msg* field is set to the ACH return code, and an exception report is generated into *payment_log* for each return file.
User creates a new payment reminder	The payment reminder is saved into the *payment_reminders* table. The *next_reminder_date* will be set to "start_date".
Run pmtPaymentReminder	1) For regular payment reminders: It finds all the entries in *payment_reminders* whose *next_reminder_date* is before than or equal to today, and sends an email for each email address. After the email is sent, the *next_reminder_date* is updated based on the *remind_interval*. 2) For check payment reminders: It finds all the checks in *check_payments* whose status is one of 7 ,8 , -1, -4 and whose *reminded* field is N. After the email is sent, the *reminded* field is changed to Y.

Table Sizing

The size to which the tables in the TBM/Payment database will grow depends on the number of enrolled users.

The following tables describe the TBM Payment tables, and the TBM tables that are related to enrollment. Most statements apply to both Oracle and Microsoft SQL Server; differences are noted. The figures in the tables assume that there are 100,000 registered users.

TBM Payment

TBM Payment Table	Projected Row Count	Notes
check_payments	1.2 million based on 100K users kept for one year	Customer dependent. Assuming one user makes one check payment per month, and check history is kept in the database for one year, then the total number of rows is approximately 12 times the number of users.
check_payments_ history	Approximately 3 times the size of check_ payments table	Tracks the state of a check payment. Usually a check passes through three states before it's cleared or returned.
check_payments_ status	10	This table is a reference to the meanings of check payment status. It is not used by TBM Payment.

TBM Payment Table	Projected Row Count	Notes
credit_card_ payments	1.2 million based on 100K users per year	Customer dependent. Assuming one user makes one credit card payment per month, and credit card payments are kept in the database for one year, then the total number of rows will be approximately 12 times the number of users.
payment_accounts	200,000	The estimated row size is approximately 1.4K per user, or 280MB for 100K users.
payment_bill_ summaries	Approximately 33K * 12 = 400K, assuming1/3 of the register with recurring payment	Saves bill information related to recurring payments.
payment_counters	< 400 per year	One row is inserted each day. Data older than one year should be backed up and deleted.
payment_invoices	12 million based on each payment averaging 10 invoices	This is usually used for business customers. Some billers may choose not to use this feature.
payment_log	< 10k per year	Approximately 20 rows will be inserted when a pmtCheckSubmit batch job is run.
payment_profile	< 100	There are approximately 20 rows for each DDN and payment type.
payment_reminders	< 100K based on 100K users	Customer dependent. Each customer can set up one reminder (each reminder creates one row in the table), but not all customers will use reminders.
recurring_payments	Approximately 33k, assuming 1/3 of the register with recurring payment	If recurring payment is turned off, then this table will be empty.

Enrollment

Table	Projected Row Count	Notes
Ach_account (nt_ach_account on SQL Server)	100k based on 100k users	Customer dependent. There is one row for each user. This table only applies to ACH.
Afcdp_account (nt_cfcdp_account on SQL Server)	100k based on 100K users	Customer dependent. There is one row for each user. This table only applies to CheckFree CDP.

Table Maintenance

For check payments, there are two tables which may grow quickly: *check_payments* and *check_payments_history*.

The *check_payments* table records the check payments made by users. The *check_payments_history table* records the history (status changes) for each check in *check_payments*. The *check_payments_history table* is approximately three times the size of the *check_payments* table.

Payments that are of a certain age (for example, one year) can be backed up and deleted from the payment database. This will ensure proper performance for TBM Payment for high volume of users. The *create_time* field in the *check_payments* table records when a check is created, and can be used to determine a check's age.

Be careful when deciding how long to keep a check in the payment database before it is removed. If you expect the number of users to be low and the database size is an acceptable size, there is no need to downsize the tables.

Backup and Recovery

All TBM/Payment database transactions operate in their own transaction context. If a single operation fails (for example, failure to enroll or submit a payment), the TBM/Payment database will be automatically rolled back to its original state.

To recover all transactions for a certain period, the database administrator should back up the database regularly so that the database can be restored to the previous day. It is best to back up the database before running the Payment Submit and Update jobs, so there will be no question about whether the jobs were still running during the backup. The frequency of backup depends on how long the period is for payment processing.

What to back up

All tables should be backed up, but the *check_payments*, *check_payments_history*, *creditcard_payments* and *creditcard_payments_history* tables in particular should be backed up on a regular basis.

Stored procedures should be backed up, especially procedures modified by edocs Professional Services.

Do not backup the master database. The master database is used by the database itself for internal purposes.

Schema

The TBM/Payment database schema is defined in the following files:

For Unix:

```
$PAYMENT_HOME/db/create_payment_schema.sql
$PAYMENT_HOME/db/alter_payment_schema.sql
```

For Microsoft Windows:>>

```
%PAYMENT_HOME%\db\create_payment_schema.sql
%PAYMENT_HOME%\db\alter_payment_schema.sql
```

Table Column Definitions

TBM Payment Tables

CHECK_PAYMENTS

This table saves check payment information.

Name	Null?	Type	Description
PAYMENT_ID	NOT NULL	NUMBER(28)	Unique for each check. It's time stamp value.
LAST_MODIFY_TIME	NOT NULL	DATE	The last time this check was updated.
CREATE_TIME	NOT NULL	DATE	The time when this check is created.
NEEDS_BACKUP	NOT NULL	CHAR(1)	Not used.
BILL_ID		VARCHAR2(255)	The ID of the bill paid by this check.
PAYEE_ID	NOT NULL	NUMBER(10)	DDN reference number.
PAYER_ID	NOT NULL	VARCHAR2(40)	User/Login ID.
PAY_DATE	NOT NULL	DATE	Check's pay date (the date the user wishes the check to be cleared).
STATUS	NOT NULL	NUMBER(2)	Check's status: scheduled(6), processed(7), paid(8), returned(-4) or failed(-1).
ROUTING_TRANSIT	NOT NULL	VARCHAR2(9)	Check's routing transit number.
ACCOUNT_NAME	NOT NULL	VARCHAR2(40)	Check account name.
CHECK_ACCOUNT_NUMBER	NOT NULL	VARCHAR2(255)	Check account number.
AMOUNT	NOT NULL	NUMBER(28,2)	Check amount.

Name	Null?	Type	Description
TXN_TIMESTAMP_1		DATE	Flexible date field. Can be used for customization.
TXN_TIMESTAMP_2		DATE	Flexible date field. Can be used for customization.
PARTITION_ID	NOT NULL	NUMBER(10)	This number is used to partition the table into fixed-size buckets for performance tuning. It is configured by Payment settings.
TXN_STATUS		VARCHAR2(20)	The transaction status returned from the payment gateway. May not be available for some gateways.
TXN_FEE		NUMBER(28,2)	The transaction fee charged by payment gateway. May not be always available.
REMINDED	NOT NULL	CHAR(1)	"Y" or "N"; indicates whether an email notification has been sent for the current status.
TXN_ERR_MSG		VARCHAR2(255)	The transaction error message returned from payment gateway. For ACH, this is the ACH return error code.
TXN_NUMBER		VARCHAR2(40)	Transaction number sent to or assigned by the payment gateway. For ACH, this is the ACH trace number.
PAYER_ACCOUNT_ NUMBER		VARCHAR2(40)	User's account number with the biller.
MEMO		VARCHAR2(255)	Check memo, which can be used for customization.
ACCOUNT_TYPE	NOT NULL	VARCHAR2(10)	Account type, payment gateway dependent. For ACH: either "checking" or "saving". For Checkfree: "DDA".
CHECK_USAGE		VARCHAR2(10)	"personal" or "business".
CHECK_NUMBER		NUMBER(10)	Check number. Not used.
ACTION_CODE		VARCHAR2(20)	Payment gateway action code. For ACH: 27(checking debit) or 37(saving debit). For Checkfree: "ADD"
LOG_ID		NUMBER(28)	This login id points to a log id in the *payment_log* table. It associates the check with a payment report.

Name	Null?	Type	Description
GATEWAY_ PAYMENT_ID	NOT NULL	VARCHAR2(255)	For Checkfree, this is the payment id assigned by Checkfree. For ACH, it matches a returned check from the ACH file to the database, when it is not possible to populate a check payment ID into the ACH file.
TXN_START_DATE		DATE	For ACH, this is the effective batch entry date for the check.
TXN_END_DATE	NULL	DATE	Reserved.
PAYMENT_SOURCE	NOT NULL	CHAR(1)	R/S: “S” means paid from UI, “R” means paid from recurring payment
FLEXIBILE_FIELD_1		VARCHAR2(255)	Flexible field for customization.
FLEXIBILE_FIELD_2		VARCHAR2(255)	Flexible field for customization.
FLEXIBILE_FIELD_3		VARCHAR2(255)	Flexible field for customization.
PID		VARCHAR2(255)	The unique id used to identify this payment account.
LINE_ITEM_ID		VARCHAR2(255)	Keeps track of data for line-item disputes.

CHECK_PAYMENTS_HISTORY

This table records the status changes that a check goes through. Whenever a check changes status, a new record is inserted into this table. A check usually goes through three statuses: "scheduled", processed" and then "paid". This means there are usually three records in this table for that check. Use this table to keep track of a check: when it is processed, when it gets paid, returned, cancelled, etc.

This table schema is exactly the same as the *check_payments* table, except that the *payment_id* is no longer a primary key.

CHECK_PAYMENTS_STATUS

This table explains the legal check payment status and their meanings.

Name	Null?	Type	Description
STATUS	NOT NULL	NUMBER(2)	Numeric value of check status.
STRING_VALUE	NOT NULL	VARCHAR(20)	String value of check status.
DESCRIPTION	NOT NULL	VARCHAR2(255)	Description of each value.

CREDITCARD_PAYMENTS

This table contains credit card payment information. TBM Payment does not save credit card numbers, so the payment gateway must have a real-time connection to a credit card processor, such as Verisign.

Name	Null?	Type	Description
PAYMENT_ID	NOT NULL	NUMBER(28)	Unique for each check. It's time stamp value.
LAST_MODIFY_TIME	NOT NULL	DATE	The last time this payment is updated
CREATE_TIME	NOT NULL	DATE	The time when this payment is created
BILL_ID		VARCHAR2(255)	The ID of the bill paid by this payment
PAYEE_ID	NOT NULL	NUMBER(10)	DDN reference number
PAYER_ID	NOT NULL	VARCHAR2(40)	User/Login ID
PAYER_ACCOUNT_ NUMBER		VARCHAR2(40)	User's account number with the biller.
CARD_HOLDER_ NAME		VARCHAR2(40)	Name on the card.
CARD_NUMBER		VARCHAR2(255)	The account number on the card. The contents depend on the payment settings.
CARD_TYPE		VARCHAR2(40)	Type of card. For example, VISA, MC, AMEX.
PID		VARCHAR2(255)	The unique id used to identify this payment account.
CARD_EXPIRE_DATE		DATE	Card expiration date.
CARD_EXPIRE_ DATE_FORMAT		VARCHAR2(40)	Format of the expiration date.
CARD_STREET		VARCHAR2(255)	Street address of cardholder.
CARD_CITY		VARCHAR2(40)	City of cardholder.
CARD_STATE		VARCHAR2(40)	State of cardholder.
CARD_ZIP		VARCHAR2(40)	Zip code of cardholder.
CARD_COUNTRY		VARCHAR2(40)	Country of cardholder.
PAY_DATE	NOT NULL	DATE	Payment date.
AMOUNT	NOT NULL	NUMBER(28,2)	Payment amount.

Name	Null?	Type	Description
STATUS	NOT NULL	NUMBER(2)	Payment status.
TXN_STATUS		VARCHAR2(20)	The transaction status returned from payment gateway. May not be available for some gateways.
TXN_ERR_MSG		VARCHAR2(255)	The transaction error message returned from payment gateway. For ACH, this is the ACH return error code.
TXN_NUMBER		VARCHAR2(40)	Transaction number sent to or assigned by payment gateway. For ACH, this is the ACH trace number.
TXN_FEE		NUMBER(28,2)	The transaction fee charged by payment gateway. May not be always available.
TXN_AUTH_CODE		VARCHAR2(40)	Transaction authentication code.
TXN_AVS_CODE		VARCHAR2(40)	Address verification code
REMINDED	NOT NULL	CHAR(1)	Determines whether a user should be sent reminder email.
PARTITION_ID	NOT NULL	NUMBER(10)	This number partitions the table into fixed-size buckets for performance tuning. The size is configurable through Payment Settings.
LOG_ID		NUMBER(28)	ID of the summary report in the *payment_log* table.
TXN_START_DATE		DATE	Transaction start date.
TXN_END_DATE		DATE	Transaction end date.
PAYMENT_SOURCE		CHAR(1)	Describes which TBM Payment function scheduled this payment. R" for recurring and "S" for single payment.
TXN_TIMESTAMP_1		DATE	For customization.
TXN_TIMESTAMP_2		DATE	For customization.
MEMO		VARCHAR2(255)	Check memo, which can be used for customization.
FLEXIBLE_FIELD_1		VARCHAR2(255)	Flexible field for customization.
FLEXIBILE_FIELD_2		VARCHAR2(255)	Flexible field for customization.
FLEXIBILE_FIELD_3		VARCHAR2(255)	Flexible field for customization.
LINE_ITEM_ID		VARCHAR2(255)	Keeps track of data for line-item disputes

CREDITCARD_PAYMENTS_HISTORY

This table records the status changes a credit card payment goes through. Whenever the payment status changes, a new record is inserted into this table. Use this table to keep track of a credit card payment: when it is processed, when it settled, returned, cancelled, etc.

This table schema is exactly the same as the *creditcard_payments* table, except that the *payment_id* is no longer a primary key.

CHECK_PAYMENTS_STATUS

Describes the possible status that a check payment can have, which is stored in the *check_payments* table.

Name	Null?	Type	Description
STATUS	NOT	NULL NUMBER(2)	Check status as a digit.
STRING_VALUE	NOT NULL	VARCHAR2(20)	Check status name.
DESCRIPTION	NOT NULL	VARCHAR2(255)	Description of check status.

CREDITCARD_PAYMENTS_STATUS

Describes the possible status that a credit card payment can have, which is stored in the *creditcard_payments* table.

Name	Null?	Type	Description
STATUS	NOT	NULL NUMBER(2)	Credit card status as a digit.
STRING_VALUE	NOT NULL	VARCHAR2(20)	Credit card status name.
DESCRIPTION	NOT NULL	VARCHAR2(255)	Description of credit card status.

PAYMENT_ACCOUNTS

This table saves information about all payment accounts.

Name	Null?	Type	Description
PID	NOT NULL	VARCHAR2(40)	Identifies this payment account.
USER_ID	NOT NULL	VARCHAR2(40)	The user who owns this payment account.
DDN	NULL	VARCHAR2(18)	The DDN name, used for ACH pre-note

Name	Null?	Type	Description
PAYMENT_TYPE	NOT NULL	VARCHAR2(10)	Either “check” or “ccard”
ACCOUNT_HOLDER_ NAME	NOT NULL	VARCHAR2(40)	The customer’s name for the payment account.
ACCOUNT_NUMBER	NOT NULL	VARCHAR2(255)	The customer’s payment account number.
ACCOUNT_TYPE	NOT NULL	VARCHAR2(40)	For check: “checking” or “saving”. For credit card: the card type, such as “visa”, "AMEX", etc.
ACCOUNT_USAGE	NULL	VARCHAR2(40)	“personal” or “business”.
ACCOUNT_STATUS	NULL	VARCHAR2(40)	Can be “active”, “inactive”, "bad_active", "pnd_active", "pnd_wait".
ROUTING_TRANSIT	NULL	VARCHAR2(9)	For check: the check routing number.
EXPIRATION_DATE_FOR MAT	NULL	VARCHAR2(20)	The date format of the credit card account expiration date.
EXPIRATION_DATE	NULL	DATE	The date when the payment account expires.
STREET	NULL	VARCHAR2(255)	Billing address.
CITY	NULL	VARCHAR2(40)	Billing address.
STATE	NULL	VARCHAR2(40)	Billing address.
ZIPCODE	NULL	VARCHAR2(40)	Billing address.
COUNTRY	NULL	VARCHAR2(40)	Billing address.
NOTIFY_SOURCE	NULL	CHAR(1)	Used for ACH prenote notification.
NOTIFY_STATUS	NULL	CHAR(1)	For ACH prenote notification. Indicates whether the payment account has been notified.
TXN_MESSAGE	NULL	VARCHAR2(255)	Contains the error message for ACH prenote or NOC.
TXN_DATE	NULL	DATE	For ACH prenote. Date of the transaction happens.
FLEX_FIELD_1	NULL	VARCHAR2(255)	Used for customization.
FLEX_FIELD_2	NULL	VARCHAR2(255)	Used for customization.
FLEX_DATE_1	NULL	DATE	Used for customization.

PAYMENT_BILL_SUMMARIES

This table saves all the bill summaries paid by recurring payments.

Name	Null?	Type	Description
BILL_ID	NOT NULL	VARCHAR2(255)	DOC id of a bill.
PAYER_ID	NOT NULL	VARCHAR2(40)	User login name.
PAYEE_ID	NOT NULL	NUMBER(10)	DDN reference
PAYER_ACCT_NUM	NOT NULL	VARCHAR2(40)	User account number with biller.
DOC_DATE	NOT NULL	DATE	Doc date of index table, when the bill was indexed.
BILL_DUE_DATE		DATE	Bill due date.
BILL_AMOUNT_DUE		NUMBER(28,2)	Bill amount due.
MIN_AMOUNT_DUE		NUMBER(28,2)	Minimal amount due.
PAYMENT_ID		NUMBER(28)	The payment id of the payment made against this bill
FLEX_FIELD_1		VARCHAR2(255)	Available for customization.
FLEX_FIELD_2		VARCHAR2(255)	Available for customization.

PAYMENT_COUNTERS

This table generates counters. TBM Payment uses this table to generate the ACH trace number, File ID Modifier, etc.

Name	Null?	Type	Description
COUNTER_NAME	NOT NULL	VARCHAR2(40)	Counter name.
COUNTER_VALUE	NOT NULL	NUMBER(28)	Counter value.
SEED	NOT NULL	NUMBER(28)	Counter start value.
INCREMENTAL	NOT NULL	NUMBER(28)	Counter incremental value.
MIN_VALUE	NOT NULL	NUMBER(28)	Counter minimal value.
MAX_VALUE	NOT NULL	NUMBER(28)	Counter maximal value.

PAYMENT_INVOICES

This table contains customer invoice information, usually obtained from TBM.

Name	Null?	Type	Description
INV_ID	NOT NULL	NUMBER(28)	Unique invoice ID generated by TBM Payment.
PAYER_ID	NOT NULL	VARCHAR2(40)	User login ID.
PAYEE_ID	NOT NULL	NUMBER(10)	DDN reference number.
PAYER_ACCOUNT_ NUMBER	NOT NULL	VARCHAR2(40)	User account number with biller.
INV_DATE	NULL	DATE	Date when the invoice is issued. Actually not used by TBM Payment and it can be customized.
INV_NUMBER	NULL	VARCHAR2(40)	A string assigned by the biller to identify this invoice. Not used by TBM Payment, so it can be used for customization.
INV_AMOUNT	NOT NULL	NUMBER(28,2)	Invoice amount.
INV_DUE_DATE	NULL	DATE	Invoice due date. Not used by TBM Payment, so it can be used for customization.
INV_ISSUER	NULL	VARCHAR2(40)	The entity that issued the invoice. Not used by TBM Payment, so it can be used for customization.
INV_MEMO	NULL	VARCHAR2(250)	Invoice memo. Not used by TBM Payment, so it can be used for customization.
INV_ISSUER	NULL	VARCHAR2(40)	The entity issues the invoice. Not used by TBM Payment, so it can be used for customization.
AMT_TO_BE_PAID	NOT NULL	NUMBER(28,2)	The actual amount being paid for this invoice.

Name	Null?	Type	Description
PROCESS_FLAG	NOT NULL	VARCHAR2(10)	This flag can be used by custom written jobs. Not used by TBM Payment, so it can be used for customization.
PAYMENT_ID	NOT NULL	NUMBER(28)	The payment id of the associated check payment.
TRACKING_NO	NULL	VARCHAR2(40)	Invoice tracking number. Not used by TBM Payment, so it can be used for customization.
TRANSACTION_ DATE	NULL	DATE	Invoice transaction date. Not used by TBM Payment, so it can be used for customization.
FLEXIBLE_FIELD_1	NULL	VARCHAR2(255)	Flexible field. Not used by TBM Payment, so it can be used for customization.
FLEXIBLE_FIELD_2	NULL	VARCHAR2(255)	Flexible field. Not used by TBM Payment, so it can be used for customization.
FLEXIBLE_FIELD_3	NULL	VARCHAR2(255)	Flexible field. Not used by TBM Payment, so it can be used for customization.
FLEXIBLE_FIELD_4	NULL	VARCHAR2(255)	Flexible field. Not used by TBM Payment and is customizable.
FLEXIBLE_FIELD_5	NULL	VARCHAR2 (1000)	Flexible field. Not used by TBM Payment, so it can be used for customization.
BILL_ID	NULL	VARCHAR2(255)	The bill id associated with the invoice.

PAYMENT_PROFILE

This table saves the Payment Settings information entered through the Command Center.

Name	Null?	Type	Description
PAYEE_ID	NOT NULL	NUMBER(10)	DDN reference number.
PAYMENT_TYPE	NOT NULL	VARCHAR2 (40)	“check”, “ccard” or “reminder”.
PARAM_NAME	NOT NULL	VARCHAR2(40)	Name of payment setting parameter.
PARAM_VALUE	NOT NULL	VARCHAR2(255)	Value of payment setting parameter.
CLOSE_DATE		DATE	Not used by TBM Payment.

PAYMENT_LOG

This table saves payment reports. Whenever checks are submitted to gateway, a summary report is generated. Whenever there is a return file from gateway, an exception report is generated. Each report contains a list of name-value pairs.

Name	Null?	Type	Description
LOG_ID	NOT NULL	NUMBER(28)	Unique ID for this report.
PARAM_NAME	NOT NULL	VARCHAR2(80)	Report parameter name.
PARAM_VALUE		VARCHAR2(512)	Report parameter value.
BATCH_INDEX	NOT NULL	NUMBER(38)	Payment internal use, record the batch indexes in a payment record.

PAYMENT_REMINDERS

This table records the payment reminders set by the users through Payment UI.

Name	Null?	Type	Description
PAYER_ID	NOT NULL	VARCHAR2(40)	Login ID.
PAYEE_ID	NOT NULL	NUMBER(10)	DDN reference.
START_DATE	NOT NULL	DATE	Date when this reminder will start.
REMINDER_ INTERVAL	NOT NULL	VARCHAR2(20)	The reminder interval: monthly, weekly, etc.
NEXT_REMINDER_ DATE	NOT NULL	DATE	The actual date the email will be sent out.
PAYER_EMAIL_ ADDR	NOT NULL	VARCHAR2(50)	User’s email address for payment reminders.
PARTITION_ID	NOT NULL	NUMBER(10)	For performance reasons, this table is partitioned into fixed-size buckets. The bucket size is configured through Payment Settings. This is the number of each bucket.
REMIND_STATUS	NOT NULL	VARCHAR2(20)	"active" or "inactive". Emails are only sent for active reminders.
USE_ENROLLMENT_ EMAIL	NOT NULL	CHAR(1)	"Y" means use the email address from the *payment_profile* table. "N" means use the email address from this table.

RECURRING_PAYMENTS

Name	Null?	Type	Description
PAYEE_ID	NOT NULL	NUMBER(10)	DDN reference.
PAYER_ID	NOT NULL	VARCHAR2(40)	User login name.
PAYER_ACCT_NUM	NOT NULL	VARCHAR2(40)	User account number with biller.
PAYMENT_ACCT_ NUM	NOT NULL	VARCHAR2(40)	For check, the check account number. For credit card, the card number.
PAYMENT_TYPE	NOT NULL	VARCHAR2(10)	“check”/”ccard”, check or credit card.
AMOUNT	NOT NULL	NUMBER(28,2)	Amount to be paid. For fixed amount, the amount specified from the UI. For pay amount due less than specified amount, the max amount specified. Otherwise, not used.
AMOUNT_TYPE	NOT NULL	VARCHAR2(40)	Can be: “fixed”, “amount due” or “less than”.
DAY_OF_PAY_ INTERVAL	NOT NULL	NUMBER(12)	Pay date: the day of the pay interval. For monthly/quarterly: 1-31. For weekly: 1-7
MONTH_OF_PAY_ INTERVAL	NOT NULL	NUMBER(12)	Applies to quarterly: 1-3
PAY_INTERVAL	NOT NULL	VARCHAR2(20)	“monthly”, “weekly”, “quarterly”.
START_DATE	NOT NULL	DATE	When the recurring payment starts.
END_DATE	NOT NULL	DATE	When the recurring payment ends.
MAX_NUM_ PAYMENTS	NOT NULL	NUMBER(12)	Maximal number of payments to be paid.
CURR_NUM_ PAYMENTS	NOT NULL	NUMBER(12)	Current number of payments have been paid.
STATUS	NOT NULL	VARCHAR2(10)	"active" or "inactive": whether the recurring payment has ended.
EMAIL_IND	NOT NULL	CHAR(1)	"Y" or "N": whether to send email when amount due is more than the amount specified.

Name	Null?	Type	Description
BILL_ID		VARCHAR2(255)	DOC id of the bill being paid, if applicable.
BILL_SCHEDULED	NOT NULL	CHAR(1)	"Y" or "N": whether the current payment has been scheduled.
LAST_PROCESS_ TIME	NOT NULL	DATE	The last time the job ran.
NEXT_PAY_DATE	NOT NULL	DATE	Pay date of the next payment available.
LAST_PAY_DATE	NOT NULL	DATE	Pay date of the last payment made.
FLEX_FIELD_1		VARCHAR2(255)	Flexible field. Not used by TBM Payment, so it can be used for customization.
FLEX_FIELD_2		VARCHAR2(255)	Flexible field. Not used by TBM Payment, so it can be used for customization.
PID		VARCHAR2(255)	The unique id that identifies this payment account.

Payment indexes

The following table lists the Indexes defined on payment and enrollment tables:

Table name	Index name	Indexed columns
payment_profile	pk_payment_profile	payee_id, payment_type, param_name
check_payments	pk_check_payments	payment_id
check_payments	nuk_check_payments_1	status, pay_date
check_payments	nuk_check_payments_2	status, reminded
check_payments	nuk_check_payments_3	gateway_payment_id
check_payments	nuk_check_payments_4	payer_id
check_payments	nuk_check_payments_5	partition_id
check_payments_history	nuk_check_payments_history_1	log_id, status
payment_invoices	pk_payment_invoices	inv_id
payment_invoices	nuk_payment_invoices_1	payment_id
payment_invoices	nuk_payment_invoices_2	payee_id, process_flag, inv_amount
credit_card_payments	pk_credit_card_payments	payment_id
credit_card_payments	nuk_credit_card_payments_1	payee_id, partition_id, payer_id, status, pay_date

Table name	Index name	Indexed columns
payment_reminders	pk_payment_reminders	payer_id,payee_id
payment_reminders	nuk_payment_reminders_1	payee_id, partition_id, remind_status, next_reminder_date
payment_log	index_payment_log_1	param_name, param_value
payment_log	index_payment_log_2	log_id
payment_counters	pk_payment_counters	counter_name
recurring_payments	pk_recurring_payments	payer_id, payee_id, payer_acct_num
reccurring_payments	nuk_recur_payment_2	status, bill_scheduled, next_pay_date
payment_bill_summaries	pk_payment_bill_summaries	bill_id
payment_bill_summaries	nuk_pymt_bill_summary_2	payer_id

Database Migration

The TBM/Payment database is designed to migrate from a previous version to new version, whenever TBM Payment upgrades the database schema. If you have a payment database from an older version of Payment, and you want to upgrade to a newer version, just run the install script that comes with the newer version. The installation script automatically alters the existing schema while preserving the old data. However, since the TBM/Payment database depends on the *document_definition_name* table from TBM, it's very important to make sure that this table is migrated, too. If a DDN name/reference is removed or changed during migration, it will cause problems for TBM Payment.

Job Scheduling

You should schedule payment jobs to run when there is not much user activity, which is typically after midnight.

If two jobs access the same table, schedule them to run at different times. pmtCheckSubmit, pmtCheckUpdate, PmtReminder pmtSubmitEnroll, pmtConfirmEnroll and pmtNotifyEnroll should run sequentially. Allow enough time between each job so that two jobs won't access the same database table at the same time. In some cases, two jobs trying to access the same table at the same time could cause a database access error.

The pmtAllCheckTasks job runs all the TBM Payment jobs sequentially. You can also edit pmtAllCheckTasks to not run specific jobs, if you wish to tailor your environment.

Appendix A: Error Messages

Job Error Messages

APP is com.edocs.services.application.LogMsgCatalog

Message ID	Severity	Message Text	User Action
APP0001	Error	Error initializing app stored procedure call strings	Call for Support
APP0002	Error	Unable to intepret the docid: {0}	Call for Support
APP0003	Exception	Exception caught: {0}	Call for support

DFE is com.edocs.tasks.statements2ir.LogMsgCatalog

Message ID	Severity	Message Text	User Action
DFE0001	Exception	StatementsToIR.getTaskParams: Failed to get task config params - {0}	Call for support
DFE0002	Exception	StatementsToIR.getTaskParams: Failed to get task config params - {0}	Call for support
DFE0003	Exception	StatementsToIR.getTaskParams: Failed to get task config params - {0}	Call for support
DFE0004	Error	StatementsToIR.setTaskParams({0}, {1}, {2}): {3}	Call for support
DFE0005	Exception	StatementsToIR.setTaskParams: Failed to set task config params - {0}	Call for support
DFE0006	Exception	StatementsToIR.setTaskParams: Failed to set task config params - {0}	Call for support
DFE0007	Exception	StatementsToIR.setTaskParams: Failed to set task config params - {0}	Call for support
DFE0008	Exception	StatementsToIR.isTaskConfigValid: Failed while validating config params - {0}	Call for support
DFE0009	Exception	StatementsToIR.isTaskConfigValid: Failed while validating config params - {0}	Call for support

Message ID	Severity	Message Text	User Action
DFE0010	Exception	StatementsToIR.isTaskConfigValid: Failed while validating config params - {0}	Call for support
DFE0011	Exception	StatementsToIR.isTaskConfigValid: Failed while validating config params - {0}	Call for support
DFE0012	Error	StatementsToIR.processTask({0}, {1})	Call for support
DFE0013	Error	StatementsToIR.processTask({0}, {1}) - Done; ivn: {2}; view: {3}; output path: {4}; validator: {5}; ir file: {6}	Call for support
DFE0014	Exception	StatementsToIR.processTask: failed to process - {0}	Call for support
DFE0015	Exception	StatementsToIR.processTask: failed to process - {0}	Call for support
DFE0016	Exception	StatementsToIR.processTask: failed to process - {0}	Call for support
DFE0017	Error	StatementsToIR.getPreviousTaskParam({0}, {1}): Failed to access {2} from previous task	Call for support
DFE0018	Error	StatementsToIR.transform: No statements for IVN {0} matched the validation criteria.	Call for support
DFE0019	Error	StatementsToIR.isValidPath: The output path is not valid. Output paths that contain only white space or are null are not allowed.	Call for support
DFE0020	Error	StatementsToIR.isValidPath: The output path, {0}, is not a directory.	Call for support
DFE0021	Error	StatementsToIR.isValidPath: The task does not have write access to the output directory, {0}, and cannot generate the IR file.	Call for support
DFE0022	Exception	StatementsToIR.isValidPath: An exception was thrown while validating the output path, {0}	Call for support
DFE0023	Error	StatementsToIR.transform: Accounts were processed for Index Volume Number {0} but there is no content in the IR file. There is mostly likely a problem with the Statement XSL stylesheet published in the Detail Extractor view.	Call for support

Message ID	Severity	Message Text	User Action
DFE0024	Error	StatementsToIR.getViewNames: Encountered an exception while accessing the view list for application {0} - {1}	Call for support

DXL is com.edocs.tasks.dxloader.LogMsgCatalog

Message ID	Severity	Message Text	User Action
DXL0001	Error	DXTableManager.removeFailedLoad: Failed to get the last loaded table name for index volume number {0}. Exception is {1}	Call for support
DXL0002	Warning	DXTableManager.removeFailedLoad: Could not expire detail catalog entry for index volume number {0} and table {1}. Exception is {2}	Call for support
DXL0003	Warning	DXTableManager.removeFailedLoad: Could not drop detail database table for index volume number {0} and table {1}. Exception is {2}	Call for support
DXL0004	Error	DXTableManager.createTable: Failed to create an entry in the detail catalog table for index volume number, {0}, and view name, {1}. Exception is {2}	Call for support
DXL0005	Error	DXTableManager.createTable: Creating database table {0} failed for index volume number {1}. Exception is {2}	Call for support
DXL0006	Information	DXTableManager.removeFailedLoad: No database table to remove for index volume number {0} and view name {2}.	Call for support
DXL0007	Error	DXTableManager.createTable: The index volume number, {0}, has already been processed for view name, {1}. This job cannot be completed.	Call for support

IAC is com.edocs.tasks.autoaccept.LogMsgCatalog

Message ID	Severity	Message Text	User Action
IAC0001	Information	AutoIndexVolAccept.processTask({0}, {1})	None
IAC0002	Information	AutoIndexVolAccept.processTask({0}, {1}): Accepting index volume: {3}	None

Message ID	Severity	Message Text	User Action
IAC0003	Error	AutoIndexVolAccept.processTask: failed to process - {0}	Call for support
IAC0004	Error	AutoIndexVolAccept.processTask: failed to process - {0}	Call for support
IAC0005	Error	AutoIndexVolAccept.processTask: failed to process - {0}	Call for support
IAC0006	Error	AutoIndexVolAccept.getTaskParams: Failed to get task config params - {0}	Call for support
IAC0007	Error	AutoIndexVolAccept.getTaskParams: Failed to get task config params - {0}	Call for support
IAC0008	Error	AutoIndexVolAccept.getTaskParams: Failed to get task config params - {0}	Call for support
IAC0009	Information	AutoIndexVolAccept.setTaskParams({0}, {1}, {2}): {3}	None
IAC0010	Error	AutoIndexVolAccept.setTaskParams: Failed to set task config params - {0}	Call for support
IAC0011	Error	AutoIndexVolAccept.setTaskParams: Failed to set task config params - {0}	Call for support
IAC0012	Error	AutoIndexVolAccept.setTaskParams: Failed to set task config params - {0}	Call for support
IAC0013	Error	AutoIndexVolAccept.isTaskConfigValid: Failed while validating config params - {0}	Call for support
IAC0014	Error	AutoIndexVolAccept.isTaskConfigValid: Failed while validating config params - {0}	Call for support
IAC0015	Error	AutoIndexVolAccept.isTaskConfigValid: Failed while validating config params - {0	Call for support
IAC0016	Error	AutoIndexVolAccept.isTaskConfigValid: Failed while validating config params - {0}	Call for support

IDX is com.edocs.tasks.indexer.LogMsgCatalog

Message ID	Severity	Message Text	User Action
IDX001	Error	Failed to load indexer library - {0}	Call for support
IDX002	Error	Indexer.getTaskParams: Failed to get field list - {0}	Call for support
IDX003	Error	Indexer.setTaskParams: Failed to set indexer config. - {0}	Error
IDX004	Error	Indexer.isTaskConfigValid: Failed while validating config. - {0}	Error

Message ID	Severity	Message Text	User Action
IDX005	Information	Indexer.processTask({0}, {1})	Information
IDX006	Error	Indexer.processTask: ddf file path invalid for job Instance: {0}, Task Order: {1}	Error
IDX007	Error	Indexer.processTask: Failed to access data file: {0} for insatnce: {1}, task order {2}	Error
IDX008	Information	Indexer.processTask: Created ir file: {0} of size {1}	Information
IDX009	Error	Indexer.processTask: Invalid count of documents processed: {0}	Error
IDX0010	Error	Indexer.processTask: ir file: {0} not accessible	Error
IDX0011	Error	Indexer.processTask: Failed to process - {0}	Error

ISC is com.edocs.tasks.ivnscanner.LogMsgCatalog

Message ID	Severity	Message Text	User Action
ISC0001	Error	IVNScanner.getTaskParams: Failed to get task config params – {0}	Call for support
ISC0002	Error	IVNScanner.getTaskParams: Failed to get task config params – {0}	Call for support
ISC0003	Error	IVNScanner.getTaskParams: Failed to get task config params – {0}	Call for support
ISC0004	Error	IVNScanner.setTaskParams({0}, {1}, {2}): {3}	None
ISC0005	Error	IVNScanner.setTaskParams: Failed to set task config params - {0}	Call for support
ISC0006	Error	IVNScanner.setTaskParams: Failed to set task config params - {0}	Call for support
ISC0007	Error	IVNScanner.setTaskParams: Failed to set task config params - {0}	Call for support
ISC0008	Error	IVNScanner.isTaskConfigValid: Failed while validating config params - {0}	Call for support
ISC0009	Error	IVNScanner.isTaskConfigValid: Failed while validating config params - {0}	Call for support
ISC0010	Exception	IVNScanner.isTaskConfigValid: Failed while validating config params - {0}	Call for support
ISC0011	Exception	IVNScanner.isTaskConfigValid: Failed while validating config params - {0}	Call for support
ISC0012	Error	IVNScanner.processTask({0}, {1})	None

Message ID	Severity	Message Text	User Action
ISC0013	Information	IVNScanner.processTask({0}, {1}) - Done; ivn: {2}; file: {3}; ir file: {4}	None
ISC0014	Error	IVNScanner.processTask: failed to process - {0}	Call for support
ISC0015	Error	IVNScanner.processTask: failed to process - {0}	Call for support
ISC0016	Error	IVNScanner.processTask: failed to process - {0}	Call for support
ISC0017	Error	IVNScanner.processTask: The number of days to scan must be a positive integer. It cannot be {0}	Call for support

IXL is com.edocs.tasks.ixloader.LogMsgCatalog

Message ID	Severity	Message Text	User Action
IXL0001	Information	IXLoader.processTask: Loaded index summary - {0}	Call for support
IXL0002	Error	IXLoader.configOK - missing param: {0}, for job {1}, ddn {2}, task order {3}	Call for support

LDR is com.edocs.tasks.ixloader.LogMsgCatalog

Message ID	Severity	Message Text	User Action
LDR0001	Exception	Loader.getTaskParams: Failed to get task config params - {0}	Call for support
LDR0002	Error	Loader.configOK - missing param: {0}, for job {1}, ddn {2}, task order {3}	Call for support
LDR0003	Information	Loader.setTaskParams({0}, {1}, {2}): {3}	None
LDR0004	Exception	Loader.setTaskParams: Failed to set task config params - {0}	Call for support
LDR0005	Error	Loader.isTaskConfigValid: Config never set for job: {0}, ddn: {1}	Call for support
LDR0006	Exception	Loader.isTaskConfigValid: Failed while validating config params - {0}	Call for support
LDR0007	Information	Loader.processTask({0}, {1})	Call for support
LDR0008	Error	Loader.processTask: ir file name: {0} is inaccessible	Call for support
LDR0009	Error	Loader.processTask: Invalid db connect info for db: {0}	Call for support

Message ID	Severity	Message Text	User Action
LDR0011	Error	Loader.processTask: The SQL loader did not generate a log file so it most likely failed to start for control file: {0}	Call for support
LDR0012	Error	Loader.processTask: The SQL loader encountered problems, see {0}	Call for support
LDR0013	Exception	Loader.processTask: failed to process - {0}	Call for support
LDR0014	Exception	Loader.processTask: failed to process - {0}	Call for support
LDR0015	Exception	Loader.processTask: failed to process - {0}	Call for support
LDR0016	Error	Loader.processTask: There are no records to process for {0}. Either the IR file has no records or the total number or records is less than or equal to the number of rows to skip.	Call for support
LDR0017	Error	Loader.getPreviousTaskParam({0}, {1}): Failed to access {2} from previous task	Call for support
LDR0018	Error	SQLServerTableLoader.modifyIRFile: The first identifiable data record in IR file {0} is an empty line. No data records will be copied to the last position in the file.	Call for support
LDR0019	Information	Loader.processTask: Loaded {0} rows successfully.	None
LDR0021	Error	TableLoader.executeTableLoad: Exception thrown while clean up temporary files...{0}	Call for support
LDR0022	Information	SQLServerTableLoader.cleanUp: Successfully removed temporary file {0}	None
LDR0023	Warning	SQLServerTableLoader.cleanUp: Unable to clean up temporary file {0}	Call for support
LDR0024	Error	SQLLoader Log: Found an error in {0} on line {1}. The line is... {2}. Please examine the sql loader log file for more details on how to correct the problem.	Call for support
LDR0025	Exception	TableLoader.validateTableLoad: Encountered a problem accessing the log file created by the sql loader... {0}	Call for support
LDR0026	Exception	TableLoader.validateTableLoad: Failed to access log file created by the sql loader... {0}	Call for support

Message ID	Severity	Message Text	User Action
LDR0027	Information	SQLServerTableLoader.cleanUp: Cannot remove temporary ir file {0} because it does not exist.	None
LDR0028	Information	SQLLoader Potential Resolution: {0}	None
LDR0029	Error	Loader.processTask: The IR file, {0}, is invalid: {1}.	Call for support

MAI is com.edocs.services.mailer.LogMsgCatalog

Message ID	Severity	Message Text	User Action
MAI0001	Error	Error intializing mailer purge stored procedure call strings	Call for support

MGR is com.edocs.services.merger.LogMsgCatalog

Message ID	Severity	Message Text	User Action
MGR0001	Error	InvalidAppException occurred while trying to access the application path: {0}	Call for support
MGR0002	Exception Error	{0} occurred while trying to access the application path: {1}	Call for support
MGR0003	Exception	Exception occurred while trying to access the input stream: {0}	Call for support
MGR0004	Exception	Exception occurred while reading versioning information: {0}	Call for support
MGR0005	Exception	C++ merger code threw an exception: {0}	Call for support
MGR0007	Error	UnsatisfiedLinkError, Check LD_LIBRARY_PATH	Call for support
MGR0008	Exception Error	Unable to deserialize the parameter properties object: {0} Unable to interpret the docid: {0}	Call for support
MGR0009	Exception	Exception occurred while application information: {0}	Call for support
MGR0010	Exception	Exception occurred while reading versioning information: {0}	Call for support
MGR0011	Exception	C++ merger code threw an exception: {0}	Call for support
MGR0013	Error	UnsatisfiedLinkError, Check LD_LIBRARY_PATH	Call for support
MGR0014	Error	Unable to interpret the docid: {0}	Call for support

Message ID	Severity	Message Text	User Action
MGR0015	Error	Exception occurred while preparing to retreive the document: {0}	Call for support
MGR0016	Error	Exception occurred while reading versioning information: {0}	Call for support
MGR0017	Error	C++ code threw an exception: {0}	Call for support

MNS is com.edocs.tasks.mns.LogMsgCatalog

Message ID	Severity	Message Text	User Action
MNS0001	Information	Started	None
MNS0002	Exception	Exception caught: {0}	Call for support
MNS0003	Information	Finished Processing	None
MNS0004	Error	Something is very wrong. Mail servers might not be working.	Call for support
MNS0005	Error	Total Accounts = {0}, Total Tried = {1}, Total Emails Sent = {2}	None
MNS0006	Exception	Exception caught: {0}	Call for support
MNS0007	Exception	Exception caught: {0}	Call for support
MNS0008	Exception	Exception caught: {0}	Call for support
MNS0009	Exception	Exception caught: {0}	Call for support
MNS0010	Exception	Exception caught: {0}	Call for support
MNS0011	Information	Created and starting...	None
MNS0012	Exception	Exception caught: {0}	Call for support
MNS0013	Exception	Exception caught: {0}	Call for support
MNS0014	Exception	Exception caught: {0}	Call for support
MNS0015	Exception	Exception caught: {0}	Call for support
MNS0016	Exception	Exception caught: {0}	Call for support
MNS0017	Exception	Exception caught: {0}	Call for support
MNS0018	Exception	Exception caught: {0}	Call for support
MNS0019	Exception	Exception caught: {0}	Call for support
MNS0020	Error	Error initializing stored procedures call strings	Call for support
MNS0021	Information	Failed to send mails, tried retry number of times, Perhaps Mail servers are not working.	Check your mail server
MNS0022	Exception	Exception caught: {0}	Call for support

MON is com.edocs.services.monitor.LogMsgCatalog

Message ID	Severity	Message Text	User Action
MON0001	Exception	Exception occurred while looking up EJB: {0}	Call for support
MON0002	Exception	Exception occurred while looking up EJB: {0}	Call for support
MON0003	Information	Monitor created	None
MON0004	Exception	Exception occurred while looking up EJB: {0}	Call for support
MON0005	Information	Monitor removed	None

PDB is com.edocs.pwc.db.LogMsgCatalog

Message ID	Severity	Message Text	User Action
PDB0001	Error	Error initializing stored procedure call strings	Call for Support

PTK is com.edocs.pwc.tasks.LogMsgCatalog

Message ID	Severity	Message Text	User Action
PTK0001	Information	Created and starting	None
PTK0002	Exception	Exception caught:{0}	Call for support
PTK0003	Exception	Exception caught:{0}	Call for support
PTK0004	Exception	Exception caught:{0}	Call for support
PTK0005	Exception	Exception caught:{0}	Call for support
PTK0006	Exception	Exception caught:{0}	Call for support
PTK0007	Exception	Exception caught:{0}	Call for support
PTK0008	Exception	Exception caught:{0}	Call for support
PTK0009	Exception	Exception caught:{0}	Call for support
PTK0010	Exception	Exception caught:{0}	Call for support
PTK0011	Exception	Exception caught:{0}	Call for support
PTK0012	Exception	Exception caught:{0}	Call for support
PTK0013	Exception	Exception caught:{0}	Call for support
PTK0014	Exception	Exception caught:{0}	Call for support
PTK0015	Exception	Exception caught:{0}	Call for support
PTK0016	Information	Finished processing task	None

PUR is com.edocs.tasks.purge.system.LogMsgCatalog

Message ID	Severity	Message Text	User Action
PUR0001	Information	{0} purged {1} records from the database	Call for support
PUR0002	Information	The task configuration for {0} has a negative purge age of {1}. No purging will occur.	Call for support
PUR0003	Error	The following exception was caught during {0} {1}	Call for support

SCH is com.edocs.pwc.scheduler.LogMsgCatalog

Message ID	Severity	Message Text	User Action
SCH0001	Information	Starting JobProcessor for ({0}) job instance: {1}	Call for support
SCH0002	Exception	Exception in processJobs: {0}	Call for support
SCH0003	Information	PWC Scheduler started	Call for support
SCH0004	Exception	Reprocessing of jobs failed: {0}	None
SCH0005	Exception	Processing of jobs failed: {0	None
SCH0006	Exception	Non-recoverable error in PWC Scheduler!: {0}	None
SCH0007	Information	Starting job instance thread for: {0}	Call for support
SCH0008	Error	Job instance initialization failed for: {0} {1}	Call for support
SCH0009	Error	Failed to get Job/Schedule Object for job instance: {0} {1}	Call for support
SCH0010	Error	Failed to update failed job status for: {0} {1}	None
SCH0011	Information	Job instance: {0}; Starting task: {1}; order: {2}	Call for support
SCH0012	Information	Job instance: {0}; Done task: {1}; order: {2}	Call for support
SCH0013	Information	Job instance: {0}; Starting task: {1}; order: {2}	Call for support
SCH0014	Error	Failed to update failed job status for: {0} {1}	None
SCH0015	Error	Failure in job processor thread for: {0} {1}	Call for support
SCH0016	Error	Failed to update failed job status for: {0} {1}	None
SCH0017	Error	Status forced to "Failed" from {0} for hung job instance: {1}	None

Message ID	Severity	Message Text	User Action
SCH0018	Error	PWC Scheduler unable to force "Fail" hung job instances: {0}	Call for support
SCH0019	Error	Failed to define next schedule for job instance: {0}	None
SCH0020	Information	Done job instance thread for: {0}	None

SCN is com.edocs.tasks.scanner.LogMsgCatalog

Message ID	Severity	Message Text	User Action
SCN0001	Error	Scanner.getTaskParams: Failed to get task config params - {0}	Call for support
SCN0002	Error	Scanner.setTaskParams - missing param: {0}, for job {1}, ddn {2}, task order {3}	Call for support
SCN0003	Information	Scanner.setTaskParams({0}, {1}, {2}): {3}	Call for support
SCN0004	Error	Scanner.setTaskParams: Failed to set task config params - {0}	Call for support
SCN0005	Error	Scanner.isTaskConfigValid: Failed to find input directory: {0}	Call for support
SCN0006	Error	Scanner.isTaskConfigValid: Failed to create output directory: {0}	Call for support
SCN0007	Error	Scanner.isTaskConfigValid: Failed while validating config params - {0}	Call for support
SCN0008	Information	Scanner.processTask({0}, {1})	None
SCN0009	Error	Scanner.processTask: Failed to create ddn volume - {0}	Call for support
SCN0010	Error	Scanner.processTask: Failed caused by an invalid input file: {0}	Call for support
SCN0011	Information	Scanner.processTask({0}, {1}): Attempting to move {2} -> {3}	None
SCN0012	Error	Scanner.processTask({0}, {1}) - attempt to move: {2} -> {3} failed	Call for support
SCN0013	Information	Scanner.processTask({0}, {1}) - attempt to move: {2} -> {3} succeeded	None
SCN0014	Error	Scanner.processTask: failed to process - {0}	Call for support
SCN0015	Information	Scanner.processTask: {0} : file length is 0.	None

SCT is com.edocs.tasks.shellcmd.LogMsgCatalog

Message ID	Severity	Message Text	User Action
SCT0001	Error	Exception caught: {0}	Call for support
SCT0002	Error	Shell Command unable to finish: {0}	Call for support
SCT0003	Information	ShellCmdTask: About to execute the following shell command: {0}	None
SCT0004	Information	ShellCmdTask: Return Value = {0}	None
SCT0005	Information	ShellCmdTask: Shell output: {0} DVN: {1}	None
SC0006	Error	ShellCmdTask: processTask: Failed to create ddn volume - {0}	Call for support
SCT0007	Error	ShellCmdTask: processTask: Unable to set task output: {0}	Call for support
SCT0008	Information	ShellCmdTask: DVN changed. Shell output: {0} DVN: {1}	None

SDF is com.edocs.services.statements2ir.LogMsgCatalog

Message ID	Severity	Message Text	User Action
SDF0001	Error	DetailTransformer.<constructor>: The application name is null.	Call for support
SDF0002	Error	DetailTransformer.<constructor>: The detail view is null.	Call for support
SDF0003	Error	DetailTransformer.<constructor>: The content writer is null.	Call for support
SDF0005	Information	DetailTransformer.transform: Cannot extract statements for a null index volume number.	Call for support
SDF0006	Information	DetailTransformer.process: The view type is null.	Call for support
SDF0007	Information	DetailTransformer.process: The view name is null.	Call for support
SDF0009	Information	DetailTransformer.process: The account resolver is not specified so all accounts will be extracted	Call for support
SDF0010	Information	DetailTransformer.process: Failed to create the XSLTMerger	Call for support
SDF0011	Information	XML2Table.getTable: Failed to parse the Detail Extractor database table definition file. The error is as follows: {0}	Call for support
SDF0012	Information	DetailTransformer.process: Failed to find the XSLT Template	Call for support

Message ID	Severity	Message Text	User Action
SDF0013	Information	DetailTransformer.process: Failed to find the DDF	Call for support
SDF0014	Information	DetailTransformer.getResolver: Failed to create the account resolver : {0}	Call for support
SDF0015	Information	DetailTransformer.writer: Failed during XSLT merger : {0}	Call for support
SDF0016	Information	DetailTransformer.writer: Data authorization exception : {0}	Call for support
SDF0017	Error	DetailTransformer.writer: Failed to read the result of the XSLT merger : {0}	Call for support
SDF0018	Error	DetailTransformer.writer: There is no content in the IR file for Index Volume Number {0}. The most likely scenario is that no accounts matched the validation criteria.	Call for support
SDF0019	Error	DetailTransformer.transform: Failed while transforming the XML to the output format : {0}	Call for support
SDF0020	Error	IRContentWriter.getHeader: The database table XML file is null or empty	Verify that the published Detail Extractor view is valid Call for support
SDF0021	Information	IRContentWriter.endProcessing: File is 0 length and will be deleted.	Call for support
SDF0022	Error	IRContentWriter.endProcessing: {0} statement(s) processed.	Call for support
SDF0023	Information	IRContentWriter.endProcessing: File is null and will be deleted.	Call for support
SDF0024	Error	XML2TableHandler.error: {0}	Call for support
SDF0025	Error	XML2Table.getTable: Failed to find a SAX Parser using the 'com.edocs.xml.sax.parser' key.	Call for support
SDF0026	Error	XML2Table.getTable: Failed to create a SAX Parser using the 'com.edocs.xml.sax.parser' key.	Call for support
SDF0027	Error	XML2Table.getTable: Errors found while parsing the create table XML. XML name: {0}.	Call for support
SDF0028	Error	DetailView.getVersionObj: Error retrieving version reader : {0}.	Call for support
SDF0029	Error	DetailView.getVersionObj: Error retrieving version set : {0}.	Call for support

Message ID	Severity	Message Text	User Action
SDF0030	Error	DetailTransformer.getAcctList: Error retrieving account list for index volume number {0}. Exception is {1}	Call for support
SDF0031	Error	XML2TableHandler.error: There is a good chance that the Detail Extractor database table description XML file does not have the following XMLSchema declaration. xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance" xsi:noNamespaceSchemaLocation="edx-DE-table.xsd".	Consult the Detail Extractor xml sample files for the correct syntax Call for support
SDF0032	Error	DetailTransformer.getAcctList: There are no accounts listed for index volume number {0}.	Call for support
SDF0033	Error	XML2Table.getTable: The XML reader {0} doesn't support one of the following features 'http://xml.org/sax/features/validation', 'http://xml.org/sax/features/namespaces', 'http://apache.org/xml/features/validation/schema'.	Call for support

SHF is com.edocs.tasks.StaticHtmlFormatter.LogMsgCatalog

Message ID	Severity	Message Text	User Action
SHF0001	Information	StaticHtmlFormatter task began processing	None
SHF0002	Error	StaticHtmlFormatter failed in the C++ code	Call for support
SHF0003	Error	StaticHtmlFormatter finished processing {0} files in time(hrs:min:sec) - {1}	None
SHF0004	Error	Exception caught: {0}	Call for support
SHF0005	Exception	Exception caught: {0}	Call for support
SHF0006	Exception	Exception caught: {0}	Call for support
SHF0007	Exception	Exception caught: {0}	Call for support
SHF0008	Error	Unable to read DDF/ALF/Template paths for this task because the version set is either missing or corrupt.	Call for support

VRS is com.edocs.services.versioning.LogMsgCatalog

Message ID	Severity	Message Text	User Action
VRS0001	Exception	Exception occurred while creating IVersionSetReader object locally: {0}	Call for support

Message ID	Severity	Message Text	User Action
VRS0002	Exception	Exception occurred whilst creating IVersionSetReader remote object: {0}	Call for support
VRS0003	Exception	Unable to instantiate remote IVersionSetReader interface: {0}	Call for support
VRS0004	Exception	Exception occurred whilst creating IVersionSetWriter object locally: {0}	Call for support
VRS0005	Exception	Exception occurred whilst creating IVersionSetWriter remote object: {0}	Call for support
VRS0006	Exception	Unable to instantiate remote IVersionSetWriter interface: {0}	Call for support
VRS0007	Exception	Exception occurred whilst creating IVersionedObj object locally: {0}	Call for support
VRS0008	Exception	Exception occurred whilst creating IVersionedObj remote object: {0}	Call for support
VRS0009	Exception	Unable to instantiate remote IVersionedObj interface: {0}	Call for support
VRS0010	Exception	Unable read data required for instantiating remote version interface implementations	Call for support

XML is com.edocs.tasks.xml.LogMsgCatalog

Message ID	Severity	Message Text	User Action
XML0001	Information	XMLFormatter task began processing	None
XML0002	Error	XMLFormatter failed in the C++ code. Return value from C++ code was: {0}	Call for support
XML0003	Information	XML Formatter task finished processing a total of {0} documents.	None
XML0004	Exception	Exception caught: {0}	Call for support
XML0005	Exception	Exception caught: {0}	Call for support
XML0006	Exception	Exception caught: {0}	Call for support
XML0007	Error	Unable to read DDF path for this task because the version set is either missing or corrupt.	Call for support

Appendix B: Glossary

Terms and Acronyms

This glossary defines 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 Telco e-Billing Manager, 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.

Addenda Record
ACH record type data needed to completely identify an account holder or to provide information concerning a transaction that carries supplemental 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 edocs' 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. Marketing Manager adds enhanced marketing promotion capabilities to TBM. 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 TBM 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. TBM 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 a TBM application) verifies the identity of an individual Web user. TBM applications must verify the identity of an enrolled user before they can view account information.

Business Logic	In TBM, 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.
Category	Subdirectory of a library. The lowest level of storage; a category has no subdirectories. A library can have one or more categories. (Term used with Marketing Manager.)
CCD	Cash Concentration and Disbursement. An automated corporate payment used primarily for the intra-company concentration or disbursement of funds. Telco e-Billing Manager supports this type of ACH payment.
Clicks	A subset of impressions and show the number of times the promotion was clicked. Term used with Marketing Manager.
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 TBM, 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 a TBM 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.
Content	What is to be shown for a promotion, such as GIF, JPEG, HTML and URLs. (Term used with Marketing Manager.)
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).

Data Store	Physical location where the content of a promotion is stored. A data store can be a database or file system and there can be one or many. (Term used with Marketing Manager.)
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 edocs' 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 TBM 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 TBM). TBM 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 a TBM application.
Detail Extractor	A type of TBM batch job used to extract and upload data from the data input file to a TBM 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 TBM 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 TBM 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 TBM 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 TBM 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 TBM 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 are: HTML, CSV, XML, Chart, XSLT, XML Query. You publish dynamic Web views.
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 TBM 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 TBM 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.
Group	A collection of tables that repeat together within a document. A group can be used to present related data elements together in moderately complex applications containing multiple or combined statement information. For example, a group could be used to consolidate shipping information with related detail in a table that repeats for each ship-to address. A group can contain tables and/or groups of tables. Also a unit that reviews and approves content before it goes into a promotion. (Term used with Marketing Manager.)
HTML Output	A type of TBM 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.)
Impressions	The number of times the page containing the promotion was accessed. (Term used with Marketing Manager.)
Indexer	A type of TBM 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 TBM extracts and presents on the Web. An input file consists of many individual documents. Also called a source file, TBM supports a wide variety of print formats, database extracts, and intermediate document composition formats. Each TBM 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	A TBM 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 a TBM 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.
Library	Storage area within a data store. There can be one or more storage areas per data store. (Term used with Marketing Manager.)
Marker	A set of delimiters defining where TBM 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 TBM, 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.
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.
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, 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 TBM 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, TBM looks to the DDF file for the extraction rules (the page style, search area and pattern) defined for the data. When TBM 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.
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. TBM supports this type of ACH 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 TBM application jobs on a regular basis to provide current account information on demand to Web users.
Profile	Defines the business logic or market segmentation behind a promotion (the basis on which a promotion occurs). Profiles are created using Composer in TBM. (Term used with Marketing Manager .)
Promotion	The specifics of a marketing campaign (the "who, what, when, where and why"). (Term used with Marketing Manager.)

Publishing	To identify a set of new or updated TBM 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 TBM 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 TBM 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.
Regular Expression	In TBM, 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.
Schedule	In TBM, when a job runs; in Marketing Manager, when a promotion runs.
Search Area	The area on a document page in an input data source, identified by start and end coordinates of rows and columns, where TBM 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.
Section	Piece of screen "real estate" on the HTML template that can be associated with conditions, to allow content of the section to vary based on data. (Term used with Marketing Manager.)
Service Provider	Business or other entity that provides online account management using one or more TBM applications.
Source file	See **Input file**.
SWIFT	Society for Worldwide Interbank Financial Transfers.

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**.
Task List	Identifies promotions that are pending approval. (Term used with Marketing Manager .)
Telco e-Billing Manager (TBM)	Edocs' Telco e-Billing Manager provides secure online account management and presentment systems capable of using a variety of input data formats.
User ID	The identification a Web-user enters to log into a TBM 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 TBM 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.
Yield	The ratio of clicks to impressions. For example, 25 clicks and 100 impressions shows a yield of 25%. (Term used with Marketing Manager.)

Index

D

R

S

T