Administration Guide

Siebel Self-Service for Cards

V5.0.1.1
Date Published: 4.18.05

Table of Contents

Preface

About This Guide

This guide is intended for system administrators and other IT professionals and describes how to install Siebel Self-Service for Cards, configure the third-party platforms that support the Siebel Self-Service for Cards production environment, and deploy Siebel Self-Service for Cards J2EE web applications.

It assumes in-depth understanding of and practical experience with system administration responsibilities, listed here.

Operating System Administration Requirements

- Start up and shut down the system
- Log in and out of the system
- Determine software patch/pack levels
- Install software & patches/packs
- Navigate the file system
- Manipulate text files
- Create files and directories
- Change permissions of files and directories
- Use basic network commands
- Transfer files with FTP
- Monitor processes & system resource usage
- Perform system backups and recovery
- Implement system security

Database Administration Requirements

- Install and configure your database server
- Start and stop your database server and database instances
- Use administrative tools

- Manage users, privileges, and resources
- Create an operational database
- Manage database files
- Manage tables and indexes
- Back up and restore databases
- Monitor database performance

Application Server Administration Requirements

- Install and configure your application server
- Start and stop your application server
- Use administrative tools
- Manage users, privileges, and resources
- Configure Java resources
- Package and deploy web applications
- Monitor application server performance

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

If you are unfamiliar with any of these tasks, please consult the related documentation for your system requirements.

Related Documentation

A PDF version of this guide is also available on your product CD-ROM.

This guide is part of the Siebel Self-Service for Cards documentation set. For more information about Siebel Self-Service for Cards, see the following guides:

Siebel Self-Service for Cards Administration Guide	How to set up and run a live Siebel Self-Service for Cards application in a J2EE environment.

Obtaining Siebel Software and Documentation

You can download Siebel software and documentation directly from Customer Central at https://support.edocs.com. After you log in, click the Downloads button on the left. When the next page appears, a table displays 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, Siebel Technical Support automatically sends you (the registered owner) an email with your license key information.

If you received an Siebel 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 Siebel. Technical Support engineers can help you install, configure, and maintain your Siebel application.

Information to provide

Before contacting Siebel 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 enables your Siebel 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 Siebel 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.
- 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 Siebel Technical Support

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

Siebel 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

Siebel 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.

Siebel escalation process has two tiers:

1. **Technical Escalation -** Siebel 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 Siebel 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 Application Setup

Before Getting Started

During the Mastering process, your project team evaluated your organization's online presentation needs along with your data input format and ETL upload requirements.

You must use the Command Center to set up and configure one or more applications to support your Siebel Self-Service for Cards installation in a live production environment.

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

- Become familiar with the data sources used by Siebel Self-Service for Cards 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 site's requirements.
- 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 Siebel Command Center requires three general steps. If you have multiple applications, it is best to set up one application at a time.

To set up a new application, you must:

1. **Create a new application.** This short step requires you to define, or name, the application in the Siebel Command Center. See "Setting Up a New Application and Jobs" on page 19.

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 maintain a production environment.

   For each job, you must choose the configuration options that will enable your application to function as intended. For some batch jobs, you must also publish associated template files.

3. **Publish files.** See”What Jobs Do I Need to Create?” on page 14 for a description of which jobs require files to be Published.

Once you have defined your application, created and configured jobs, and published any required files, you can proceed to set up a schedule for each job and begin live production. Note that the command center does not automatically schedule jobs to run; you must manually specify job schedules for production. See “Scheduling Jobs” on page 36.

What Jobs Do I Need to Create?

Each command center application requires certain batch jobs run on a recurring basis to maintain current data.

The specific number and type of production (batch) jobs you need to create and configure depends on the type of input file your application use.

You may need to create and configure a combination of the following jobs for an application:

- **ETLDataUpLoad job**, which loads data into the database so it can be accessed by Siebel Self-Service for Cards.
- **EpicWareDbLoader**, which loads the account routing numbers received from Epicware.
- **SystemReport job** creates reports about user activities.
- **Hierarchy Import and Hierarchy Synchroniztion jobs** only if you use the hierarchy manager feature and need to import an existing hierarchy to your system.
- **Various Payment jobs**, which are described starting on page 51.

Production Jobs

EpicWareDbLoader

Loads routing numbers that Siebel Self-Service for Cards can use to validate account information when customers enroll. For more information, see “EPICwareDbLoader” on page 21.

This job should be run monthly, after the EpicWare CD with new account information arrives.

Purge Logs Job

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

For more information about the Purge Logs job and other system maintenance activities, see Command Center online help and the *Payment Gateways* section on page 53.

For details on setting up a Purge Logs job, see “Creating and Configuring a Purge Logs Job” on Page 29.

cmDelayedResponseHandler

When a request is sent to the FDR, FDR may not send the response in a timely manner (within the time out period) due to system problems. When this happens, the FDR adaptor throws a transaction time out exception and stores the information about the request in a database table. For example, if a user makes a payment or direct payment, a check is inserted to the *check_payments* table with the status set to -99 (undefined). The delayed response from FDR may arrive sometime later, and will be available in the FDR queues.

This job examines events in the FDR queue. If it finds payment messages, the job updates the payment tables with the details (the status is either 7 (‘processed’) or -1 (‘failed’)), and deletes that entry from the FDR queue.

Schedule this job to minimize the number of failed payments that would have succeeded when the problem was resolved. For example, if you know that a link will be down at a certain time, schedule this job to run after the link comes back up.

No configuration is required in the Command Center.

You should run the cmDelayedResponseHandler job daily.

ETLDataUpLoad

This job processes the Delta, MSR and Monetory files from FDR, and uploads them into the database. For configuration and operational details, see “ETLDataUpLoad Job” on page 21.

ETLPurge Job

This job cleans up ETL related data from the OLAP database. For more information, see page 25.

ScheduledMessagesJob

Some events cause Siebel Self-Service for Cards to send email immediatly; other events cause email to be scheduled. Email that is scheduled is sent by the ScheduledMessages Job.

To configure, edit the XML configuration file *EDX_HOME/modules/messaging/messaging.xma.xml*, which dynamically updates the system. The following properties should be edited to specify how messages are sent and who it says they come from::

```
 <bean id="config" class="com.edocs.common.messaging.config.ConfigBean">


  <property name="from"><value>administrator@somecomp.com</value></property>
  <property name="retry"><value>12</value></property>
  <property name="smtpHost"><value>EXCHANGEUS</value></property>


     </bean>
```

No job configuration is necessary in the Command Center. This job should be run daily.

SystemReports

Generates various reports for Siebel Self-Service for Cards users, both internal (CSR) and external (customers). For configuration and operational details, see “System Reports” on page 31.

NotificationSynchronizer Job

The NotificationSynchronization job cleans up staging tables used by the ETL job, and failed notifications. For more information, see “ETL Purge” on page 25.

What Files Do I Need to Publish for a New Application?

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 required by batch jobs

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 publish them all at once in bulk.

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

Files you publish for batch jobs

Batch Job Type	**Files you are instructed to publish as part of the job configuration:**
Report	A ZIP archive containing reporting related files.

Steps to Configure Command Center for Siebel Self-Service for Cards

The following steps in the given order need to be executed for the Command Center to be setup to be used with the Siebel Self-Service for Cards Product.

1. Setup payment gateways as described in the section Payment Gateways

2. Create a new DDN with the name "ReportApp" and using the Publisher, publish the reporting-1_0-publish.zip file located in the root directory of your Siebel Self-Service for Cards installation.

3. Run the EPICwareDbLoader job to load payment routing information (this step can be ignored if no actual data is required. A sample set of routing numbers is uploaded as part of the sample application setup for CM)

2 Setting Up a New Application and Jobs

Logging Into the 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 Siebel Self-Service for Cards.

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

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

To log into the Command Center:

1. Verify that the Web server and the database server are both running.
2. Launch your Internet browser.
3. Enter the URL for the Command Center servlet configured when Siebel Self-Service for Cards 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 Siebel Self-Service for Cards does not recognize the ID and password, consult your system administrator or the person who installed.
5. Click Submit. Siebel Self-Service for Cards 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 Command Center.

   Click Create New Application from the Main Console. The command center displays the Create New Application screen.

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

3. Enter the JNDI name of the datasource EJB to use for this 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 application. 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. You can ignore the Index Partition Count.

5. Click **`Create Application and Continue`**. The Create New Job screen displays.

6. Proceed to create and configure jobs for your application.

About mapping your application to a datasource EJB

You must specify a datasource EJB for each application (DDN) you create in the Command Center. When creating an 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 Siebel Professional Services representative. The packaged EJB which can be used by default to setup a job has a JNDI name of *edx/ejb/edxDataSource*.

EPICwareDbLoader Job

Loads routing numbers that Siebel Self-Service for Cards can use to validate account information when customers enroll. This job should be run monthly, after the EpicWare CD with new account information arrives.

Number of Record to Skip - Specifies how many records to skip in the beginning of the DAT file. The first record of *Epicware.dat* contains the record count.

Maximum size in the bind Array - Specifies the number of maximum records to be inserted into the specified table.

Batch Size of Rows to Load - Specifies the records to be inserted and committed as a batch.

Amount of Error Allowed in insert - The number of errors to tolerate in insertions. The data upload process will be aborted when the threshold specified is reached.

Additional Control Parameters for SQLLDR - Specify the control parameters for the loader. The parameters are specified in the sqlldr manual. Note that the parameters are **not** validated.

Path of the Data file to be loaded - Specify the location and filename of the EpicWare data file, usually called e*picware.dat*. Note that using wildcards is **NOT** supported.

Output Path for the EpicWare DataLoader- The directory where the EPICwareDbLoader job creates the log file and other control files. This directory **MUST** exist.

When the job is complte, the *epicware_data Table* will be populated with the records needed to upload.

Caution

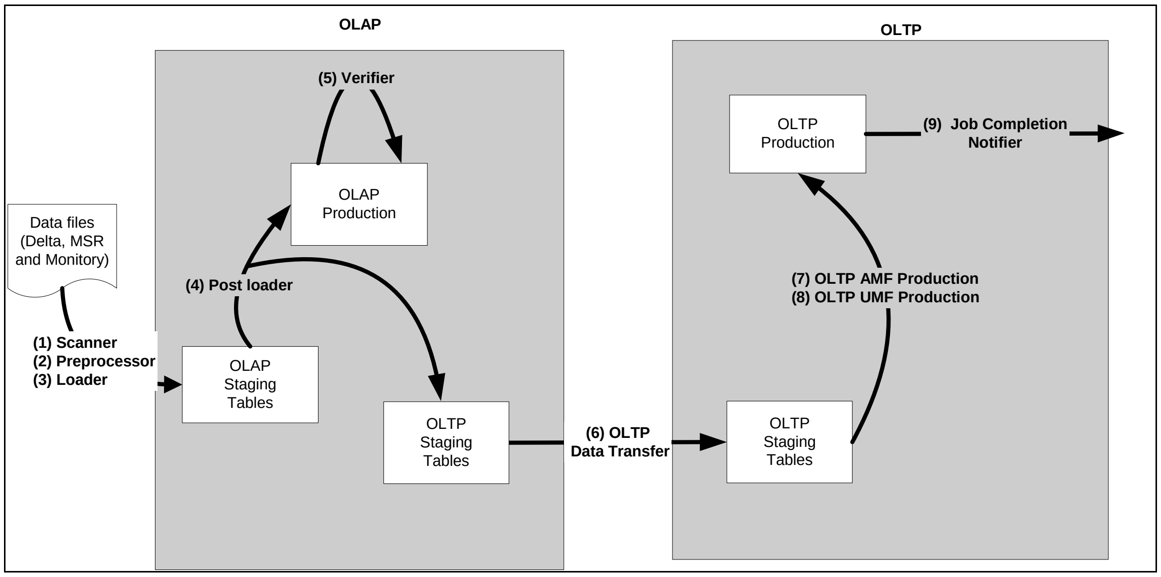
If the app server is run as any other user other than root , then that user must either have ORACLE_HOME, ORACLE_BASE and LD_LIBRARY_PATH in their PATH, or source WL_HOME/common/commEnv.sh in that user's .profile to run EPIC ware job successfully.

ETLDataUpLoad Job

The ETLDataUpLoad job processes the Delta, MSR and Monetory files and uploads them into the database. All three file types must be uploaded in the given order (Delta, MSR and Monetory). These files update the card member account numbers and bank accounts for new, updated or stolen credit cards.

Edit the *edx_etl_sqlldr.config* file with the information for the OLAP instance.

The ETLDataUpLoad job's sub tasks are shown in the following diagram, and described following the diagram:



Caution: You can use symbolic links for the Input and Data directories as long as they are under one file system

Edit the *edx_etl_sqlldr.config* file with the information for the OLAP instance.

Important: You MUST run the **Notification Synchronizer** job after running the ETL job

Caution If any FDR file fails to process, you must request a new file from Metris. Specifically, if you see "Record Unknown" errors in the preprocessor task.

Task 1: ETLFileScanner

Scans the files only if all three files have been received from FDR. If any of the three input files does not exist; this task goes to the No Operation state. Not only must all files be present, but they can not have zero bytes. If there are zero byte files, the job will fail in the preprocessor.

Note that even though there can not be zero bytes, they can have zero records. Here is an explanation of zero byte and zero record files:

- Zero byte file means - the size is just zero byte;. the file does not have any data in it. It is an empty file. It is just a file system entry.
- Zero records means - the file has no records. Still the file has a header and sometimes a trailer too. This adds to the file size. Thus the size of a MSR file that has zero records has to be at least 8000 bytes. The size of a zero record Delta file is 2*189 = 378 bytes. The size of zero records monetary file is hard to tell; it will be in the vicinity of 250 bytes.

Task 2: ETLPreprocessor

The processor formats each file into the specified database load format. The preprocessor creates nine files for each MSR file. Two files are created for each Delta file. One file is created from each Monetary file. For each corresponding incoming data file, this job creates an XML file and a preprocessed file.

If all the files are successfully processed, then they are copied to the Archive File Path.

Task 3: ETLOLAPCleanUp

Performs clean up of MSR and MON data, which is remaining in the system as a result of a previously failed ETL job.

Task 4: ETLDataLoader

Delta, MSR and Monetary data are loaded to the corresponding staging tables in the OLAP database.

Task 5: ETL PostLoader

Verifies the data loaded by ETLData Loader, and moves the data from the OLAP staging to OLAP production. OLTP staging tables are also populated in the OLAP database schema.

OLTP staging uses the following tables:

- edx_rpt_amf_stage
- edx_rpt_umf_stage
- edx_rpt_eapay_stage
- edx_rpt_amf_dscl_stage
- edx_rpt_ntf_cust_acct_stage
- edx_rpt_ntf_stmnt_stage
- edx_rpt_ntf_unbill_trans_stage

The verification process writes soft and hard errors to the *edx_etl_rule_error_log* table. If soft errors occur, the task will continue. But if hard errors occur, then the task fails. Notifications are sent to the administrator for each error. The administrators name and the email address is configured in the job.

Task 6: ETLVerifier

(Future - Verifies that the data move from the staging to the Production tables in the OLAP database was successful.)

Currently, this does nothing. Verification for Metris is done by ETLPostLoader (previous) task.

Task 7: ETLOLTPDataTransfer

Transfers the data in the OLTP staging tables (which reside in the OLAP database) to the identical OLTP staging tables in the OLTP database.

Task 8: ETLOLTPProduction

Runs the following sub-processes:

OLTP AMF Production - Loads or updates the OLTP production table(s) data from the “edx_rpt_amf_stage” staging table. The following production tables will be updated:

EDX_AMF_ACCOUNT
PAYMENT_ACCOUNTS
CHECK_PAYMENTS
CREDITCARD_PAYMENTS

OLTP AMF DSCL Production - Loads or updates the OLTP production table(s) data from the “edx_rpt_amf_dscl stage” staging table. The following production tables will be updated:

```
EDX_AMF_ACCOUNT
```

OLTP UMF Production - Loads or updates the OLTP production table(s) data from the “edx_rpt_umf_stage ” staging table. The following production tables will be updated:

EDX_UMF_CONTACT_PROFILE

OLTP Payment Production - Loads or updatse the OLTP production table(s) data from the “edx_rpt_eapay_stage” staging table. The following production tables will be updated

PAYMENT_ACCOUNTS

Task 9: ETLUnbilledTranPurge

Purge all unbilled transactions for which a statement is available.

Task 10: ETLEventDispatcher

Fires an ETLCompleteEvent signifying that ETL job has completed, which triggers notification handlers. These handlers send the notifications for the user and account Payment updates and for statement notifications.

ETLPurge Job

The ETL Purge job handles cleaning up of the OLAP ETL related tables. It has two command center tasks:

- Statement Purge Task
- Unbilled Transaction Purge Task

Statement Purge Task

There should be a mechanism to delete old statement data from the OLAP statement tables to regain database space.

Configuration in Command Center

The configurable settings for this task are:

- **Number of months to retain statements:** Indicates the number of months to retain statement history.
- **Purge staged data**: **Y** causes staged data to be deleted along with the statement data. **N** causes only statement data to be purged.

If this job completes succesfully, then the qualifying records from the following OLAP tables will be deleted:

Relevant partitions will be dropped from following tables,

EDX_RPT_BILL_CYCLE_DIM
EDX_RPT_STATEMENT_FACT
EDX_RPT_BILLED_TRANS_FACT
EDX_RPT_MERCHANT_TRANS_LINK
EDX_RPT_CATEGORY_TRANS_LINK
EDX_RPT_FINANCE_CHARGE_FACT
EDX_RPT_FLAP_PROMO_FACT
EDX_RPT_REWARD_FACT
EDX_RPT_CUST_STMT_MSG_FACT
EDX_RPT_TRANS_MEMO_FACT

Relevant rows will be deleted from,

EDX_RPT_FEED_FILE_FACT
EDX_RPT_ETL_PART_DRIVER_DIM

The Following database tables get Inserted or updated.

EDX_RPT_PURGE_LOG

Unbilled Transaction purge Task

The unbilled transactions uploaded through ETL job must be deleted so these transactions will not appear on the statement as billed after they are valid.

Configuration in Command Center:

No configuration is required.

If this job completes succesfully, then the qualifying records from the following OLAP tables will be deleted:

Records are deleted from EDX_RPT_UNBILLED_TRANS_FACT.

The Following database table is updated.

EDX_RPT_ETL_PART_DRIVER_DIM

NotificationSynchronizer Job

The NotificationSynchronization job re-invokes unsuccessful handlers, cleans up staging tables used by the ETL job, and invokes notification handlers for queued ETLComplete events. This job should be scheduled to run every day within time period in order to recur in time intervals.

To configure this job, specifiy the administrator name and email address.

This job's tasks perform the following functions:

- **FailRecoveryTask -** Recalls unsuccessful/failed handlers by invoking them again.
  - If all the handlers have a status of SUCCESS in *edx_ntf_hnd_process_status*, then the HandlersSynchronizerTask will run (see the description under HandlersSynchronizerTask).
  - Notifies NotificationHandlersController about the existence of unsuccessful handlers (which means the handlers are in INITIAL status) by sending a NotificationRecoveryEvent. Then NotificationHandlersController will try to invoke any handlers which are in initial status. Also notifies the administrator with the list of currently uncompleted handlers.
  - If there are any handlers in FAILED status, NotificationSynchronization notifies the administrator about failed handlers and this job fails. Administrative involvement is required.
- **HandlersSynchronizerTask -** Notifies the administrator about any failed/unrecoverable handlers, and purges the OLTP stage tables.
  - Purges the data in the OLTP stage tables.
  - If there are any “queued” ETLComplete events, it resends the oldest event, in order to invoke NotificationHandlersController, which in turn calls handlers to serve that ETL cycle.

Troubleshooting NotificationSynchronizer

Purging of the staging tables happens based on the date the ETL job runs, which must match the created date of the staging data. Staging data could exist for multiple dates. If you do not have any data created on the same date that is found in the *edx_ntf_hnd_process_status* table, then data will not be purged.

If you have data with a different date than what is in the *edx_ntf_hnd_process_status* table, and there is any entry in the *edx_ntf_hnd_etl_queue* at the time Notification sync job runs, then that data will be purged during the next run (as soon as all the handlers are done with it).

Some common problems are:

- The Notification-Synchronizer job runs, but I don’t get e-mails.
- When I run both ETL and Notification-Synchronizer jobs, I get a warning mail saying that Handlers are still running!

First check that the following conditions are met:

- To test this job, you must enroll a user that has your e-mail address.
- Check that the messaging module XMA configuration file *messaging.xma.xml* has the SMTP server setup correctly.

Background

An account must meet certain conditions to receive e-mail. For example, in CM for Metris, the following e-mail rules are supported:

[A1] A new statement is available AND e-mail has not been sent for the current cycle (batch, MSR).

[A2] Payment is past due AND enrolled in Account Central (batch, CHD)Current Date is greater than the Payment Date

[A3] Payment due within x days (batch, CHD, once per statement cycle): If Current Date - Payment Due Date < X System creates a batch e-mail notification

[A4] Credit limit is reached (batch, CHD):Current Balance meets or exceeds card member’s Credit Limit

[A5] Balance is within $X of Credit Limit (batch, CHD, once per statement cycle):If Current Balance - Credit Limit < 0 AND an e-mail has not been sent already for this statement cycle.

[A6] Balance exceeds x amount (batch, CHD, once per statement cycle):If Current Balance > X AND an e-mail has not been sent already for this statement cycle.

[A7] Balance drops below x amount (batch, CHD):If Current Balance < X AND an e-mail has not been sent already for this condition,

[A8] Payment posts to account (batch, MON):System finds payment posted to Account from MON file upload

[A9] Credit posts to account (batch, MON):System finds credit posted to Account from MON file upload

[A10] Transaction exceeds x amount (batch, MON): transaction for > $X for Account

Batch notifications are triggered after the ETL job completes. For example, when the ETL job uploads Delta, MON, and MSR files for a given statement cycle. Batch Notification/Alerts and event handlers are driven by the ETL complete event, which is sent by the final task of the ETL job. In the CM for Metris there are 10 event handlers for the notification types listed previously. Each of these event handlers acts based on certain business condition evaluations; if the condition is met, then a Notification/Alert is sent.

There is no easy way to monitor these handlers, which run as daemon processes; they must complete their task without failure. To ensure this, we use a recovery mechanism implemented in a Command Center job called “NotificationSynchronizer”.

This job includes two tasks:

- FailRecovery task
- HandlersSynchronizer task

The FailRecovery task restarts all the fail handlers. A failed handler is indicated by the status column in the database table *edx_ntf_hnd_process_status*. The status can be INITIAL, SUCCESS, or FAIL. Initially, all the rows of that table have a ‘status’ of INITIAL. When when the handlers complete successfully, the value for all rows is SUCCESS. If a handler fails, the value is either FAIL or INITIAL. The status stays INITIAL if there is a catastrophic failure, because the events handlers cannot update the status table.

If the FailRecovery task runs and sees a handler in the INITIAL state that is not running, then it restarts that handler, and sends an admin ‘info’ alert saying that that event handler is still runnng. The subject of the admin ‘info’ alert e-mail is “Handler Synchronizer Notification”. If there is at least one event handler which Notification & Alert system cannot recover automatically, then manual/administrator intervention is required. Handlers in the INITIAL state continue to run.

Each time a handler restarts it decrements a retry count, which is defined in the *notification-handlers-info.xma.xml* file in the property name ‘maxTries’. This value is stored in the *edx_ntf_hnd_process_status* column ‘remaining_tries’ when the system starts. If the value reaches zero , its state is set to FAIL.

If the FailRecovery task detects a FAIL status for an event handler then the NotificationSynchronizer job fails.

The HandlersSynchronizer barrier-synchronizes all the Notification events handlers. If all Notification event handlers are in the SUCCESS state, then the HandlersSynchronizer task purges *edx_ntf_hnd_process_status*, and all the staging tables that involved in Notification and Alerts. In the CM for Metris there are five staging tables in OLTP schema.

Note that staging tables are purged based on the 'date' the ETL complete occurs. This date is stored in the *edx_ntf_hnd_process_status* table when events handlers are run.

Note also that Notification and Alert system assumes that two ETL jobs never run on the same date. It checks whether another ETL complete event has occurred by checking the *edx_ntf_hnd_etl_queue* table. If finds an ETL complete event, then it resends the events, which causes the system to start sending notification/alerts for the next cycle.

You can run the NotificationSynchronizer any time. If there is no data in notification staging tables then nothing happens, and the job goes to no-op. It will start again in the next recurring/scheduled time.

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.

Caution

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

Once you run a Purge Log job, the data deleted will no longer be available for inclusion on View Log reports.

To create and configure a Purge Logs job:

1. On the Main Console, click the application name, listed under Applications in the table. The Edit Application screen appears.
2. Click **`Add New Job`**. The Create New Job screen displays.
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`**. The Purge Logs configuration screen displays.
6. Specify the configuration parameters for the PurgeLogs task.
7. When finished, click **`Submit Changes and Schedule`**. Siebel Self-Service for Cards submits the job configuration parameters and displays the Schedule screen. You can schedule the Purge Logs job later.
8. Click **`Main Console`**.

PurgeLogs Task

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

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

Creating and Configuring a Purge Logs Job

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

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

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`**. The Create New Job screen displays.
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`**. The Purge Logs configuration screen displays.

6. Specify the configuration parameters for the PurgeLogs task:

Value	Description
0	Purges all information up to midnight of the current date
1	Purges all information up to midnight of the previous day
>1	Purges all information up to midnight of the day specified.

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

8. Click Main Console.

PurgeLogs Task Configuration Options

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

PurgeLogs Task Configuration

Field	What to enter/select
Purge Prior to (Number of Days)	Specify one of the following values to determine what to purge: **0** - Purges all information up to midnight of the current date **1** - Purges all information up to midnight of the previous day >**1** - Purges all information up to midnight of the day specified.

System Reports

The SystemReport job generates several types of reports, chosen through job configuration.

Inputs:

- **Report Type**: Choose the type of report to create.
- **No of Hours** – Chose the period that this report should cover: the last 1, 24 or 48 hours.
- **Output Directory -** Path of the directory where the reports will be written.
- **Report List -** Chose the type of report to generate. Report types are described in the following table.

Report List	Title	Description
CARMEM_ACT	Cardmember Activity Report	Tracks online Cardmember activity including enrollment events, login event, log out or timeout events, view statement, submit an NLS query, and send a secure message.
CARMEM_ADD_CHG	Address Change Report	Provides a list of all accounts where potential fraud may be occurring. The change types tracked are same day as enrollment, same day as forgot password, and same new address as another address change today or during previous day
CARMEM_LOCKOUT	Cardmember Lockout Report	Lists Cardmember users that have become locked out due to login and registration failures.
CARMEM_PRO	Cardmember Profile Change Report	Lists Cardmember user profile changes, which include all Address fields, Home Phone Number, Business Phone Number, Email Address, Password and Language Indicator
CSR_ACTIVITY	Internal Users Activity Report	Serves to detect fraudulent activity of the Metris Internal Users who have access to card member data. The report tracks Internal User activity in the Internal User application and actions made on behalf of Cardmember (tracking the same information as the Cardmember Activity Report). Internal User activity includes searching Cardmember by SSN, jumping to page, resetting password, unlocking Cardmember, unlocking account number, unlocking Internal User, changing Internal User Status (active or inactive) and changing Cardmember user status (active or inactive).
CSR_LOCKOUT	Internal User Lockout Report	Lists Internal users that become locked out due to login failures and password expiration.
CSR_PRO_CHG	Internal User Profile Change Report	Lists all changes to the Internal user’s profile. A change is defined as addition, deletion, and update, where applicable, to Case Management categories, roles, privileges, escalation, status (active or inactive), password, and language indicator.
CSR_ROLE	Internal User Role Report	Lists all Internal users and their roles and privileges. Internal User statuses tracked are active or inactive only.
CSR_SECURITY_VIO	Internal User Security Violation Report	Tracks failed login attempts by internal admin users and the reason for the failure.
EMAIL_NOTIFICATION	E-mail Failure Report	Lists e-mails that have been sent by AccountCentral.
FILE_RECON	File Reconciliation Report	The File Reconciliation Report tracks the number of statements (primary keys) processed.
PAY_DIRECT	Direct Bill Pay Report	Tracks all Cardmember activity for the Direct Bill Pay Program.
PAY_ONLINE	Payment Report	Tracks all Cardmember daily online payments.

Report List	Title	Description
SECURE_MSG	Secure Messaging Report	Shows case management productivity for all Internal users. Case activity status include opened, accepted, escalated, closed and declined.
USER_INFO	User Information Report	Shows enrollment and log in report. This report will report on enrollments, logins and statement views for the given day (from midnight to midnight)

The rest of the parameters to this job select the range of data to use when creating the report.

Creating and Scheduling the SystemReports Job

This section explains how to create a system report job to run it hourly (Metris requirement).

1. Create a new application from the Command Center
    - Provide a name for a Application in Application Name field
    - Use the Datasource Name **`edx/ejb/EdocsDataSource`**
    - Keep the default value for Index Partition Count
2. Create and configure a new job

    Step 1:

    - Provide a name for the job in Job Name field
    - Select Job Type **`System Reports`**

    Step 2, Publish application/job files and templates.

    - In the Publisher click **`Create`**, select the **`Bulk Upload`** tab. Then click **`Browse`** to locate the **`reporting-1_0-publish.zip`** file, then click **`Upload`** to publish it.

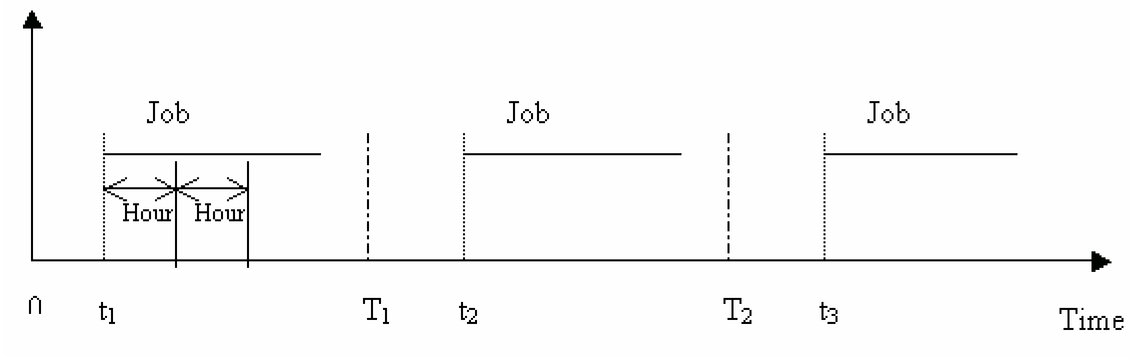
    Step 3 Configure and Schedule job.

    - Click on Configure Job and Continue.

    Command center displays the job configuration screen.
3. Configure the BRFReportGenerator task.
    - Select the configuration parameter **`No of Hours`** value of “1” from the drop down.
    - Provide a file system path for the Output Directory. This is where the generated system reports are stored for FTP access.
    - Select the hourly report type to generate from the Report List.
    - Click **`Submit Changes`** and **`Schedule`** buttons.

4. Schedule the job to run hourly:
   - Specify a valid start date for the schedule to take effect.
   - Specify the schedule window parameters and any repeating parameters

   The Schedule window is the time period that the job takes to run in each recurring attempt. The hourly scheduling falls during this window. This time window has a start-time and an end-time:



   Here the job is repeated for the schedule intervals 0, T1, T2, … and the Job starts in each interval at time t1, t2, t3, … respectively. The time "window" is: for a given time interval, which is how long the job runs.

   For hourly jobs, you mustconfigure retry for hourly within this time window.

   If the Job is to run hourly for 24 hours, then the time window is 24 hours.

   The Job should recur in intervals 0, T1, T2, … the setting for recurring is show below for the job to recur every day. You could also set up the job to recur until a given date.

   You may wish to setup Holidays on the Holiday Settings page. If you define Holidays, the system reports will not be generated on those holidays.

5. Save the schedule or click **Run now**.

This configuration causes the job to run as scheduled to generate reports hourly that are stored in the file system.

3 Managing the Live Production Process

General Production Monitoring Activities

Once you have set up and configured an application and its jobs, you use the Siebel 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; see "Scheduling Jobs" on page 36.
- Monitoring the status of jobs and individual production tasks; see "Monitoring Production Jobs" on page 38, "Viewing Job Status" on page 40, and "Viewing and Verifying Task Status" on page 40.
- Starting a job; see also "Canceling and Retrying Failed Jobs" on page 42.
- Monitoring system services; see "Monitoring System Services" on page 45.

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 44.

General system maintenance activities:

- Run activity reports to review application usage statistics.
- Maintain the database. See “Database Administration” on page 47.

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
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.
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).

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 Siebel 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

Job Schedule Parameters	
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 5 ▼ 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).

Tip 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 42.

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.

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
Mail Notification	Check the job status window. Verify emails sent.

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.

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 the application server.
3. Display the Service Status again to verify that the problem has been corrected. If services are still missing, refer to your Installation Guide.

4 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 the database

Changing the Administrator's Password

You can change the Administrator's password, which you use to log into the Siebel 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.

Holiday Settings

Holiday Settings allow you to define the holidays used by XRS and its modules when processing requires information about holidays. For example, Payment can not schedule a payment on a holiday.

There are options to Create, Copy, Edit or Delete a calendar.

Creating a Calendar

After selecting **`Create New Calendar`**, you can select the year for which to set holidays by clicking the arrows surrounding the year at the top of the page.

Then select the days you wish to make holidays by clicking on them. A day is selected as a holiday when it is shaded.

When you are done adding holidays, click on **`Save Calendar`**.

Changing the Password to Real Time FDR

Update the property values in the file */opt/edocs/xma/modules/connectors/fdrConnector.xma.xml* with the same values you specified in *fdrProperties.properties*. Then restart the WebLogic server instance of cmuser. Here is an sample file:

```
<property name="queueManager">
               <value> MEQA </value>
         </property>
         <property name="inQueue">
               <value> FDR.ODS.MQ4R.QUEUE </value>
         </property>
         <property name="outQueue">
               <value> FDR.ODS.RQST.QUEUE N</value>
         </property>
         <property name="user">
               <value>EDOCSID</value>
         </property>
         <property name="password">
               <value>GR8SK8S</value>
         </property>
```

Changing logger settings

By default the logging is set to write to the file system. The location where logging is written and the levels of logging (error, debug, info etc.) and can be changed by editing the *<EDX_HOME>/modules/logging/edx-logging-config.xml*.

By default, all log files are created in the same directory as the *startWebLogic.sh* or *startManagedWebLogic.sh* scripts.

To change the logging to be done to the database uncomment the lines below:

```
<!--
   <default-handler name="DEFAULT_JDBC_HANDLER"
class="com.edocs.common.logging.handlers.JDBCAppender">
         <param name="dsprovider"
value="com.edocs.common.logging.handlers.DataSourceProvider"/>
         <param name="jndi" value="edx.logger.databasePool"/>
         <param name="buffer" value="25"/>
   </default-handler>
-->
```

And and comment the following:

```
   <default-handler name="DEFAULT_FILE_APPENDER"
class="org.apache.log4j.FileAppender">
         <param name="Append" value="false"/>
         <param name="File" value="logging.log"/>
   </default-handler>
```

To change the logging level change the highlighted text:

```
<global>
<level-ref ref="error"/>
<handlers>.........
</global>
```

Level	Description	Value in xml
Error	Logs only Errors	error
Error and Warning	Logs Errrs and Warnings	error_warn
Log and Error	Logs information and errors	log_err
All	Logs Information, Debug, Warning and Errors	all

5 Payment Jobs

pmtCustom

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

pmtEPICWareDbLoader job

The EPICware database contains all valid FDR account numbers, which Siebel Self-Service for Cards uses to verify account numbers before initiating communications with FDR. This database of account numbers is updated by running the pmtEPICWareDbLoader job.

The Epicware CD will be received once a month. This CD contains a single data file (*Epicware.dat*), which is manually copied to the appropriate input directory. After copying the file, schedule the job to update the database.

The pmtEPICWareDbLoader job completly refreshes the *Epicware_Data* table in the OLTP database that is used by Siebel Self-Service for Cards.

Parameters are:

Number of Records to Skip in the Beginning - Specify how many records to skip in the beginning of the DAT file. The first record of *Epicware.dat* contains the record count.

Maximum size (in bytes) of the bind array (Maximum Size) - Specifies the number of maximum records to be inserted into the bind array table.

Batch size of Rows to Load - Specifies the number of records to be inserted as a "batch" and "committed".

Amount of Errors Allowed in inserts - The number of insertion errors to tolerate before aborting the data upload process.

Additional Control Parameters for SQLLDR - Specify the control parameters to the loader "for", as specified in the sqlldr manual. These parameters are not validated.

Path of the Data file to be Loaded (fullpath) - Specify the path and filename of *epicware.dat*.

Output Path for the EpicWare DataLoader - Specify the location for the log file and other control files created by the epicware data loader.

pmtPaymentStatusNotify job

Notifies users about the status of their payment(s). For example, if a user schedules a check payment (to pay some or all of the credit card balance), this job sends notification that a check payment has been scheduled.

pmtPaymentStatusNotify Configuration

Check payment settings are for scheduled ACH payments to the Metris credit card balance. Credit card payment settings are for Direct Bill Payments.

The configurable parameters for this job are:

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

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

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

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

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

Payment Job Scheduling

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

6 Payment Administration

Payment Gateways

FDR Payment Gateway

FDR is a third party gateway which is used for submitting check payments. To use the payment module the FDR gateway must be set up. This is done by clicking the Settings link on the left tool bar in the command center and then clicking the Payment Settings link.

This gateway is used for all the check payments made through the Siebel Self-Service for Cards application.

FDR Payment Gateway Configuration

You must create a DDN called “eapay”. The FDR gateway should be configured on for this DDN. The FDR gateway parameters are:

Instant enroll check accounts with cassette – This should be always “Y”. This option allows the bank account to immediately enroll with FDR and be available to the user for making payments without any delays.

FDR System number – This contain the FDR system number that is given by FDR.

Principle user number – This contains the Principle User Number given by the FDR

Terminal ID – This contains the Terminal ID provided by FDR.

Operator Code – This contains the Operator Code specified in FDR

FDR file output directory – This contains the directory path for the output files of FDR

FDR External Payment Gateway

FDR External payment gateway is used for paying external payees using the Metris credit card. This gateway is used for submitting all the external payments to the FDR. You must create a DDN called “eapayexternal”.

FDR External Payment Gateway Configuration

The configuration of this is also similar to the FDR gateway except all fields are optional. After configuring the gateway the DDN which was used to create the FDR External Gateway should be registered with the **global configuration tab** located in Payment Settings page. You need to select the appropriate DDN where the FDR external payment gateway was configured.

Payment Database

Preventing Multiple Payments

By default, 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 Payment is the same as the *doc id* in the XOD (oltp1) database.

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, 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 Payment performance. Make sure to create the proper index if you plan to create a new query.

Payment Table Sizing

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

The following tables describe the Payment tables, and the XOD 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.

Payment

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 Payment.
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 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.

Caution 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 XOD/Payment database transactions operate in their own transaction context. If a single operation fails (for example, failure to enroll or submit a payment), the XOD/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 Siebel Professional Services.

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

Schema

The XOD/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
```

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

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
MGR0008	Exception Error	Unable to deserialize the parameter properties object: {0} Unable to interpret the docid: {0}	Call for support

Message ID	Severity	Message Text	User Action
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
MGR0015	Error	Exception occurred while preparing to retreive the document: {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

Message ID	Severity	Message Text	User Action
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

Message ID	Severity	Message Text	User Action
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

Message ID	Severity	Message Text	User Action
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
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

Message ID	Severity	Message Text	User Action
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

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
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

Message ID	Severity	Message Text	User Action
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