Troubleshooting Guide for Siebel Billing Manager

Version 5.1
Date Published: 2.28.2006

ORACLE | SIEBEL

Contents

1 Preface

2 General Troubleshooting Process

How to Begin Troubleshooting 7

Verifying Product Version 8

Log Files 8

Interpreting Exceptions 12

Using Debug 14

Reviewing Recent System Changes 17

3 Job Failure

Multiple Job Failures 19

When Jobs Hang 19

When the System Hangs 20

Indexer Job Failure 20

EmailNotification Job Failure 24

4 DDF Validation Errors

Fixing Invalid DDF Content 31

5 Improving Application Performance

Summary List Response Time 35

Indexer Job Performance 35

EmailNotification Job Performance 36

1 Preface

About This Guide

This guide is intended for anyone who administers or maintains the proper functioning of Siebel products at their organization, and describes how to initiate the troubleshooting process, identify critical information about what was happening in your system and applications when the problem occurred, and suggests ways to resolve the problem.

2 General Troubleshooting Process

How to Begin Troubleshooting

If you have a problem with your application, there are several steps you should take to get the troubleshooting process underway.

Please open a Siebel SupportWeb Service Request if you are unable to resolve the problem using the guidelines in this book. Assist your support representative by having important system information at hand to enable them to more quickly resolve the problem.

Follow these general steps to troubleshoot a problem:

1. Determine the customer impact; if the problem is in production, is batch processing or statement presentment affected? How many customers are affected (100% or a few)? Are payments affected?
2. Verify the product version and patch level.
3. Check whether you have a customized build of the product and whether there have been any recent changes made.
4. Use the Command Center to verify job and task status and examine the error message.
5. Establish when the problem started and how often it occurs. Did it ever work? Did a dependent system fail, like db or an integration point? Does it happen in other environments? Has anything changed or been added since it last worked?
6. Look at application and system logs for error or exception detail. Are all customers seeing this error? How many exceptions are in the log? When did the first exception occur? Is there a pattern?
7. Use the information in this guide and the Siebel SupportWeb Knowledge Base to interpret exceptions or errors and to find possible causes and solutions, in particular, see:
   - The release notes for your version to check for known issues.
   - The appropriate section in this guide on troubleshooting a job failure.
   - The Support Search option on the Siebel SupportWeb to search the Knowledge Base for details about your problem.
8. Make any necessary corrections or changes.

9. Try to rerun the job or resume normal processing.

Verifying Product Version

It is important to know the version of your Siebel product and the versions and patch levels of your platform to:

- Look up specific troubleshooting information about your product and version in the Siebel SupportWeb Knowledge Base.
- Provide your Siebel support representative if you contact Siebel Technical Support and open a service request.

Verify and provide version and patch levels of:

- WebLogic
- WebSphere
- Operating System
- Database Server
- Database Driver
- Middleware (MQ Series)
- Your Siebel product

Verify the patches required in the Siebel Release Notes for your product are correctly installed.

To determine which Siebel product version you have:

- View the content of the version.txt file in your root *EDX_HOME* installation directory.

(The Siebel version also indicates patch level.)

Log Files

Whenever a problem arises, review the Billing Manager application logs along with your system logs for details about any errors that occurred and what activity was taking place at the time.

Billing Manager

Billing Manager maintains logs of all activities that occur and messages generated during production. You can use the Command Center to create a report showing the following types of log messages generated over a select time period:

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

It is a good idea to review these logs on a regular, ongoing basis to monitor jobs in your production environment.

Billing Manager 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 Billing Manager task where the error occurred and the level of error.
Message	Message text.

To view Billing Manager production logs:

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. Billing Manager displays the log messages of the selected type generated during the selected date and time range.
8. To select different log information to view, click **`Reselect Log View`**.

You can also view this information directly in the Billing Manager system_activity database table.

WebSphere

WebSphere maintains a set of logs to record various activity, error, and output information.

WebSphere Log File Name
native_stderr
native_Server_stdout
ServerName.pid
startServer.log
stopServer.log
Systemerr.log Systemerr_*.log
Systemout.log Systemout_*.log

Location of WebSphere Logs

WebSphere generates the logs to %WAS_HOME%/logs, where %WAS_HOME% is the directory specified when the server instance was created in WebSphere (for example, $APP_SERVER_HOME\AppServer\logs*servername*).

WebLogic

WebLogic maintains a set of logs to record various system event data.

WebLogic Log File Name	Type of Information
weblogic.log	All console output for the server
access.log	Record of files accessed from the server.

Using weblogic.log and access.log

Timestamps indicate the start and end of a log entry.

In weblogic.log, look for entries with <error> or <exception>; you can ignore <notice> and <info> entries.

Access.log contains the IP of the person who accessed the page, the URL of the page they accessed, and HTTP errors. An error 200 appears if it was successful; other error numbers may indicate a problem.

For example, in the following access.log entry, the HTTP status code is 500, which is a server error:

```
10.xxx.xxx.x71 - - [21/Oct/2003:15:29:19 -0700] "GET
//Payment/Payment?app=Payment&ddn=acme&acctNum=8744&form=payCr
edit&amountDue=406.67&billId=ivn-3592/po-17334386/bc-2230/pc-
1/dd-20030912 HTTP/1.1" 500 0
```

If you got a 500 error there must be an exception in the weblogic.log. Look for an exception in weblogic.log from around this time, such as:

```
####<Oct 21, 2003 3:29:19 PM PDT> <Info> <EJB> <server1>
<myserver> <ExecuteThread: '12' for queue: 'default'> <> <>
<010051> <EJB Exception during invocation from home:
com.edocs.ps.acme.DBAccessBean_33lj8w_HomeImpl@7bdfd0 threw
exception: java.rmi.RemoteException: EJB Exception:; nested
exception is:

   java.rmi.RemoteException: ORA-01403: no data found

ORA-06512: at "EBPP.ACME_DBACCESS", line 133

ORA-06512: at line 1

>
```

For a description of HTTP error codes, see:

http://www.w3.org/Protocols/HTTP/HTRESP.html

You can also use access.log to generate a chronological list of actions taken by the user. Sorting the entries by IP yields the sequence of steps the user took that generated the error or problem.

Be sure to look at the access.log for the customer-facing server.

Location of WebLogic Logs

WebLogic generates the logs to %WL_HOME%/config/mydomain/logs, where %WL_HOME% is the directory specified when the instance was created in WebLogic (and mydomain is as you have configured).

Solaris

To create a log, you must add a nohup command to your WebLogic or WebSphere startup scripts. You specify the directory in the nohup command.

Solaris Log File Name	Type of Information
nohup.out (or other name you specify at startup)	Debug information

On WebLogic, stdout and stderr information goes to nohup.out if you start WebLogic via nohup (a process immune to hangups) with no redirection to an output file, for example (using Bourne shell syntax):

```
% nohup startWebLogic.sh &
```

Having a single nohup.out file isn't very practical, however, because it gets overwritten every time WebLogic restarts, and is also not a very functional name. To maintain a history of log files, redirect the output from nohup to a log file using an appropriate naming convention. For instance, using Bourne shell syntax, to launch WebLogic with stdout and stderr information going to console_weblogic_admin_server.log, use the command:

```
% nohup startWebLogic.sh > console_weblogic_admin_server.log
2>&1 &
```

It is also a good idea to append the log file name with the timestamp of when WebLogic was started.

Interpreting Exceptions

An exception is an error that the application or system generates when it encounters a problem while running in a live environment. You must correct the problem before processing can resume normally. Exceptions can appear in:

- The Siebel error log
- J2EE server logs. These paths are configurable in the J2EE console.
- The nohup.out log for debugging Solaris
- The Windows console (when not running as a service for debugging)

It is possible for a Siebel exception to get captured in the server log only, so be sure to check all relevant logs. See "Log Files" on page 8 for details. When an exception occurs, it generates a Java call stack trace in one or more logs. A stack trace is a list of raw audit information about what was happening when the exception occurred. The most recently executed code is at the top and usually contains the error message of interest.

Sample stack trace:

```
Servlet failed with IOException
java.rmi.RemoteException: EJB Exception: ; nested exception is:
.javax.ejb.EJBException: java.rmi.RemoteException:
java.sql.SQLException: ORA-01001: invalid cursor
ORA-06512: at "EDX_DBA.Acme", line 148
ORA-06512: at line 1
javax.ejb.EJBException: java.rmi.RemoteException:
java.sql.SQLException: ORA-01001: invalid cursor
ORA-06512: at "EDX_DBA.Acme", line 148
ORA-06512: at line 1
.at
com.edocs.services.application.AppIndexVolMgr.getHitListByIvn(AppIndex
VolMgr.java:850)
.at
com.edocs.services.application.AppIndexVolMgr_fcksga_EOImpl.getHitList
ByIvn(AppIndexVolMgr_fcksga_EOImpl.java:348)
at com.edocs.app.verify.Verify.getHitList(Verify.java:584)
.at com.acme.util.RejectProcessor.getAccounts(Unknown Source)
.at com.acme.util.RejectProcessor.processRejects(Unknown Source)
.at com.acme.util.RejectProcessor.<init>(Unknown Source)
.at
jsp_servlet.__accountnumberlist._jspService(__accountnumberlist.java:1
32)
.at weblogic.servlet.jsp.JspBase.service(JspBase.java:27)
.at
weblogic.servlet.internal.ServletStubImpl.invokeServlet(ServletStubImp
l.java:265)
.at
weblogic.servlet.internal.ServletStubImpl.invokeServlet(ServletStubImp
l.java:304)
.at
weblogic.servlet.internal.ServletStubImpl.invokeServlet(ServletStubImp
l.java:200)
.at
weblogic.servlet.internal.WebAppServletContext.invokeServlet(WebAppSer
vletContext.java:2546)
.at
weblogic.servlet.internal.ServletRequestImpl.execute(ServletRequestImp
l.java:2260)
.at weblogic.kernel.ExecuteThread.execute(ExecuteThread.java:139)
.at weblogic.kernel.ExecuteThread.run(ExecuteThread.java:120)
```

To troubleshoot an exception:

1. Look at the stack trace and identify some basic information. The top few lines of the call stack usually contain the most important information, such as:

   - **Whether it's an Oracle (SQL), SQLServer, DB2, or a Java error.** In the sample exception above, it's an Oracle error, not Java error, so it originated in the database.
   - **Whose code is the problem.** If you see **`com.edocs`** the problem is related to Siebel code and you can look it up in the SupportWeb.
   - **Description and other information.** You may see a brief description of the error. In the sample, you see "invalid cursor." (EDX_DBA.Acme is the package name.)

2. Search the Siebel SupportWeb Knowledge Base for the exception number. The Knowledge Base is the most up-to-date repository of specific troubleshooting information. The Siebel support staff maintains the Knowledge Base on a regular basis with resolutions to issues.

If the exception occurred due to a Java problem, it indicates the name of the class file. The top Java method shown is the one that threw the exception. If necessary, your Siebel Technical Support engineer can trace a stack trace back through the methods to the related code to determine the cause of the problem. Be sure to have the exception and stack trace information at hand if you contact Siebel Technical Support. It is best to attach this to the case or, if brief, paste it into the case body.

Using Debug

About debugging

Setting debug keys for various system features enables debugging, letting you trace and expose specific code-related information about what is occurring in your Siebel software, including input parameter values used to call a method, results, etc.

Each debug flag is designed to debug a particular area of functionality where you're having a problem, such as EmailNotification, user enrollment, or presentment, and starts debugging whenever that feature is executed.

For example, if a stack trace for a crash mentions libformatter, setting a debug flag can show which view it was trying to display. This information could help determine whether a data file or ALF was deleted, the ALF path is incorrect, if there is a problem with read permissions, or if there is some other related issue.

Running with debugging enabled can slow system performance.
Some debug flags are too low-level to be useful for debugging.
Debug message content varies in format and may be of limited use to non-programmers or for general troubleshooting purposes.

You can insert debug parameters in the startWeblogic script or any customized scripts you may be using.

Using Java flags

To set debug keys using Java flags:

On WebSphere:

1. Add debug keys to the system properties in the JVM Settings tab for each server in the WebSphere console. For example:

   ```
   Name = com.edocs.payment.debug
   ```

```
Value = true
```

2. Restart the servers.

On WebLogic:

Windows

Debugging does not produce an output in Windows when Billing Manager is running as a service. You must run Billing Manager from the command line while you are in debug mode. This "logs" the output to the console on your desktop. You can also redirect that output to a file.

- It is best to pass the debug Java flag in edx.config.bat. For example:

```
Set JAVA_OPTIONS="$JAVA_OPTIONS -DMAIL_DEBUG_KEY=1"
```

UNIX

- In edx.config:

```
JAVA_OPTIONS="$JAVA_OPTIONS -DMAIL_DEBUG_KEY=1"
EXPORT JAVA_OPTIONS
```

Debug keys you can set as Java flags

Billing Manager Debug Key	Use to debug…
-DJAVA_ONLY	Bypass core classes to compose test HTML statements (to establish whether the problem is in Java or core classes).
-Dcom.edocs.fs.logging.System.out=true	Sends all log messages to System.out rather than JMS.
-Dcom.edocs.app.chart.debug=true	Billing Manager Charting
-Dcom.edocs.tasks.mns.debug=true	EmailNotification Job
-Dcom.edocs.mns.db.debug=true	EmailNotification Job
-Dmail.debug=true	EmailNotification Job
-Dcom.edocs.core.debug=true	JNI
-Dcom.edocs.services.application.debug=true	Application (DDN related)
-Dcom.edocs.services.mailer.debug=true	Email issues
-Dcom.edocs.services.merger.debug=true	Bill presentment
-Dcom.edocs.services.reporting.debug=true	Reports
-Dcom.edocs.services.versioning.debug=true	Version set/views
-Dcom.edocs.pwc.debug=true	Command Center
-Dcom.edocs.pwc.tasks.CommonTask.debug=true	Command Center

Billing Manager Debug Key	Use to debug…
-Dcom.edocs.enrollment.user.debug=true	User enrollment
-Dcom.edocs.enrollment.user.edocs.debug=true	User enrollment
-Dcom.edocs.enrollment.access.debug=true	User enrollment
-Dcom.edocs.app.publisher.debug=true	Publisher
-Dcom.edocs.app.user.debug=true	Presentment
-Dcom.edocs.services.session.debug=true	ISession
-Dcom.edocs.pwc.db.debug=true	Scheduler
-Dcom.edocs.tasks.staticHtmlFormatter.debug=true	StaticHTMLFormatter task
-Dcom.edocs.tasks.ixloader.debug=true	IXLoader task
-Dcom.edocs.tasks.indexer.debug=true	Indexer task
-Dcom.edocs.tasks.ivnscanner.debug=true	IVNScanner task
-Dcom.edocs.tasks.shellcmd.debug=true	Shell commands
-Dcom.edocs.tasks.scanner.debug=true	Scanner task
-Dcom.edocs.payment.debug=true	General Payment functionality
-Dcom.edocs.payment.encrypt.debug=true	Payment for encryption-related issues
-Dcom.edocs.eapost.debug=true	General post functionality

Note: You can enable/disable some of these values using a deployment descriptor.

Setting as an environmental variable

To set debug keys as an environment variable:

For all platforms, it is a good idea to set debug flags environment variables in the Billing Manager configuration file, edx.config.

To set the debug flag as an environmental variable in UNIX:

- Set the following environmental system variable, as:

  ```
  EDOCS_DUMP_JNI_ARGS=1
  export EDOCS_DUMP_JNI_ARGS
  ```

To optionally set the debug flag as an environmental variable in Windows:

Add the debug key to edx.config.bat:

1. Add a new line at the top right, after the “Main:” line, and add the key, as in:

   ```
   SET EDOCS_DUMP_JNI_ARGS=1
   ```

2. Restart the server.

Debug keys you can set in the environment

Billing Manager Debug Key	Use to debug…
EDOCS_DUMP_JNI_ARGS=1	Input and output of the libformatter.so

Using Log4j

We recommend Log4j, an open source tool that can simplify logging. It inserts log statements into your code, a low-tech method for debugging. Log4j lets you enable logging at runtime without modifying the application binary. The log4j package is designed to include log statements in shipped code without incurring a heavy performance cost. See logging.apache.org for more information.

Reviewing Recent System Changes

Find out what changes, if any, have been made to your system recently, including:

- Upgrades or point-releases
- Any new features
- Any changes to related software or hardware
- Customizations to your application made in-house or by Siebel Professional Services
- Using Siebel SupportWeb Knowledge Base

The Siebel SupportWeb Knowledge Base is the most up-to-date repository of specific troubleshooting information about Siebel Self-Service products. The extensive Knowledge Base contains thousands of articles and is continually updated by the Siebel Technical Support staff.

3 Job Failure

Multiple Job Failures

If multiple jobs fail at once:

1. Find out whether the database is down. This is the most likely cause.
2. Look for a possible job "collision." If you are running multiple Indexer or EmailNotification jobs, turn on debug.

   We do not recommend scheduling more than one of these jobs to run at one time. It's a better idea to configure Indexer to run multiple job instances concurrently; see the Command Center help for more information.

When Jobs Hang

If one or more jobs hang (versus fail), and the entire server is not hung, try the following steps to recover:

1. Stop scheduler using:

   ```
   wl_scheduler -stop url t3://audi:7001
   ```

   Where *audi:7001* is the name and port number of your production server. When scheduler stops, the jobs that were running appear as Failed.
2. Shell command tasks launched by Command Center have their own process ID, and therefore may continue to run. You need to kill these processes separately.
3. When scheduler restarts, all jobs set to run in the past run at once. This can cause conflicts. Change the schedule before restarting.
4. Restart scheduler using:

   ```
   wl_scheduler -start url t3://audi:7001
   ```

When the System Hangs

If the server hangs, follow these steps to help identify the problem:

1. Look at the CPU usage. Are you at %100?
2. Create a series of thread dumps to see what the CPU is working or waiting on. You can capture the thread dump in UNIX with:

   $ **kill -3** ***pid***.

   Where pid is the Java process ID. This sends the thread dump to std.out. Do four of these in a row, 30 seconds apart, and send the logs to Siebel Technical Support.

Indexer Job Failure

General steps to troubleshoot Indexer job failure

An Indexer job fails when its status is Failed or No operation.

Follow these general steps to gather basic information about the failure and resolve it:

1. View task status to find which task failed; click the status of the job (`Status` column) on the Main Console.
2. View the Billing Manager Error log to see if Billing Manager generated an error or exception. Also see “Interpreting Exceptions” on Page 12. (The error number prefix indicates which task was processing when the error occurred: SCN, IDX, IXL, IAC, etc.)
3. View your server and other system logs to see if any errors were generated to these files.
4. Look for specific solutions:
   - Check the resolutions in this chapter to problems related to the particular task where the Indexer job failed.

     For further information, the *Siebel Billing Manager Administration Guide* describes the function of each task in detail, including input, output, and configuration parameters.
   - Search the Siebel SupportWeb Knowledge Base on a specific error, exception, or descriptive text that appears in a log. (You can always make a more general search on the Knowledge Base as well.)

5. Retry the failed Indexer job after correcting the problem (unless you're using Payment).

When to retry vs. cancel an Indexer job

When an Indexer job fails it is always safe to retry the job (unless you're using Payment). If the problem is resolved and the job can proceed, Indexer resumes where it left off. For example, if the job failed during IXLoader, the Indexer job continues with that task since it doesn't need to scan the data file again.

You can cancel an Indexer job, if necessary.

Scanner task failure

If the Indexer job fails at the Scanner task (status is Failed or No operation), here are some possible causes:

Type of Failure	Probable Causes	What to Do
Task fails (with No Operation status)	Data file missing or not accessible.	Check the input directory for a valid data file.
	Wrong input directory specified.	Check the Indexer job configuration for the specified Input directory and edit the pathname if necessary. If you get error SCN0014 Scanner.process.Task on Solaris, search the Knowledge Base for FAQ 121. (Solaris Data directory for the indexer job may be incorrect and mounted under /var/crash or there may be symbolic links associated with it. Remove the symbolic links and create a new data directory under /opt/Siebel/EDX_HOME, and ensure it is referenced correctly in the Indexer job.)
	Wrong output directory specified.	Check to make sure all directories exist, have the correct permissions and are mounted. Search the Knowledge Base for FAQ 218 or "Invalid task config" message for Post.
	Permissions do not allow read access.	Check the directory and data file permissions, correcting permissions if necessary.
	Two servers (Scheduler) are accessing the same database.	Don't run Scheduler on two servers that are both accessing the same database for job configuration information (such as a production application on one server and a test application on another server). Run one Scheduler at a time or replicate the database and file structure.
	Scheduler not running.	Verify whether Scheduler (Process Workflow Controller, PWC) is running and restart it if it is

Type of Failure	Probable Causes	What to Do
		not. Scheduler is a java process; verify that this process is running. **Solaris/AIX**: Check ps-ef |grep xboot (You can check the java processes, but it's tough to tell which java process is the wl_scheduler ... especially on a server with multiple WebLogic servers running.) % cd $EDX_HOME/bin % ls –la look for a file named .scheduler.localhost:8010.pid where <localhost:8010> is the –url argument passed when launching the wl_scheduler and either "cat" or "more" this file to reveal the PID for the wl_scheduler process. Then verify if this PID is running via "ps –ef |grep <*PID*>" **Windows**: Check command prompt window or in the list of services (if you ran it as a service). Check the pwc log file for messages. When you start Scheduler, the system creates a log file in EDX_HOME%\Logs. Note that if the Server, Scheduler, and Logger processes are not running, the server may be down. See your *Siebel Billing Manager Installation Guide* for details about starting/restarting Scheduler.
	The Oracle database is down.	Check whether the database is up and restart the edx_db database instance. Stop and start WebLogic and retry the Indexer job.
	Oracle Listener is stopped.	Make sure the Listener process is running.
Task fails (with Failed status)	Out of disk space.	Check disk space and free up if necessary.
	File has 0 bytes.	Check the size of the file data file in the Input directory; if it is 0 bytes, this file is the problem. Indexer will not process zero byte files. Scanner assumes a zero byte file is corrupt and the task/job fails.

Indexer task failure

If the Indexer job fails at the Indexer task, check for any of the following potential causes:

Type of Failure	Probable Causes	What to Do
Task fails (with Failed status)	Corrupt DDF or data file.	Check against known good files if possible. Replace the corrupted file. To verify the data in an input file can be read, open the file in DefTool and simulate using the application DDF. If the files are being sent to the server in binary mode and you receive error message GE0629 on AIX or Oracle, search the Knowledge Base for FAQ 142.
	Permissions do not allow read access.	Check the directory and data file permissions, correcting permissions if necessary.
	Path to DDF not found.	Check the Indexer task configuration. If the DDF path says "Not found," go to Publisher and verify that the version set was published properly (with the correct DDF and type). Check that EDX_HOME is properly set in startup files and that EDX.CONFIG is sourced
	Out of disk space.	Check disk space and free up if necessary. Check swap space using df-K. Add more space if necessary.

IXLoader task (also DXLoader) failure

If the Indexer job fails at the IXLoader task, check for any of the following potential causes:

Type of Failure	Probable Causes	What to Do
Task failed (Failed status). No *.ctl or *.log files were created in the Data directory. It appears as if SQLloader didn't run.	Database communication problem.	A "SQLLoader.processTask" error may indicate a problem with the configuration of the database connection. Check the Oracle tnsnames.ora file on the application server, and try to manually connect to the database via sqlplus, TOAD, etc. Is Oracle Listener Started on the database server? Check Listener.ora. Is SID correct? Check dbmap, dbaccess.properties (the files are from the 1.9 days – no longer around in 2.x and greater.)
Task failed, a partial .ir file was	Out of memory.	Verify min and max heap sizes for java in $*EDX_HOME*/config/edx.config. Also verify

Type of Failure	Probable Causes	What to Do
created, but no *.log or *.ctl files.		min and max heap sizes for java, which are specified as JAVA_OPTIONS in the scripts that launch the WebLogic servers: startWebLogic. sh and startManagedWebLogic.sh. Check swap space using df-K. If swap space is filled add more then retry the job.
Task failed, but *.log and *.ctl files were created.	Bad *.ir file (or *.ix file)	Check *.log and *.ctl for SQLloader error messages.
	Out of disk space.	If you receive an error such as IXL0015, verify that the application server has sufficient swap space and increase as necessary.
	Pointing to wrong database.	If a valid data file exists and the permissions on the folder allow read access, this is usually a mistake in pointing to the incorrect database. Make sure the JDBC and SQLLoader are both pointing to the same database.

AutoIndexVolAccept task failure

The AutoIndexVolAccept task does not typically fail.

EmailNotification Job Failure

General steps to troubleshoot EmailNotification job failure

An EmailNotification job fails if an email message fails to send to the SMTP mail server (after retrying the job the number of times specified in the “Max Number of Retries” field in the job configuration).

When an EmailNotification job fails, some emails have typically already been successfully sent to the server. But the failed email and subsequent email remain unsent, and remain unsent until you resolve the problem email and rerun the job. Until you cancel a failed EmailNotification job, no other files can be processed by the EmailNotification job. (Any sent mail does NOT get sent again when you retry a job.)

When the problem is resolved, you must retry the job to send the unsent email.

Follow these general steps to troubleshoot EmailNotification job failure:

1. Check the task status to verify which task failed; click the status of the job (`Status` column) on the Main Console.

2. View the Billing Manager Error log to see if Billing Manager generated an error or exception. See “Interpreting Exceptions” on Page 12.

3. View your server and other system logs to see if any errors were generated to these files.

4. Check the configuration parameters set for the EmailNotification job in the Billing Manager Command Center.

5. Look for specific reasons for email failure:

   - When EmailNotification fails it generally means troubleshooting the failure of individual email. For reasons why email may fail and how to correct it, see the MailNotification Task Failure section below.

     For further information, the *Siebel Billing Manager Administration Guide* describes the function of the IVNScanner and MailNotification tasks in detail, including input, output, and configuration parameters.

   - Search the Siebel SupportWeb Knowledge Base on a specific error, exception, or descriptive text that appears in a log.

6. Once you’ve corrected the problem, retry the failed EmailNotification job. If it fails again on other problem email, you must resolve the specific problem with that email before you can retry the job. You must resolve each email failure since the rest of the unsent email won’t get sent otherwise.

When to retry vs. cancel an EmailNotification job

If you have corrected the problem with a failed email, connection to the server, or other problem that caused an EmailNotification job to fail, you can always retry the job.

Canceling an EmailNotification job prohibits you from retrying the job, and any unsent messages cannot be sent for that data file.

There is a workaround for restarting a cancelled job, however, it requires you to update the database directly. Knowing how to perform this workaround enables you to cancel the failed job so that you can start another mail job while researching and correcting the problem with individual email failures, then later restart the cancelled job so that the unsent email can be sent.

How to update the database to force a retry of a cancelled EmailNotification job:

1. Set the status of the failed email to “address error” in *DDN*_mail table for the accounts and IVN that failed.

2. In Siebel SupportWeb, search for FAQ 221. This article contains a link to the eNotificationReset.sql script, which you must run to reset the IVN for mail notification. This script deletes the row from ddn_volume_types where z_ivn = *your ivn* AND volume_type = 'IVN Scanner email-Email Notification.' This will allow the IVN to be picked up again next time the email job runs.

   Edit the following section of the script to replace "ddn" with the application name, and "&2" with the number of the cancelled IVN you want to rerun:

   ```
   delete
   from   ddn_volume_types
   where  ddn_vol_num = &2
   and    volume_type = 'IVN Scanner email-Email Notification'
   ```

3. Run the EmailNotification job.

IVNScanner task (for EmailNotification) failure

IVNScanner does not typically fail.

MailNotification task failure

Type of Failure	Probable Causes	What to Do
Task failed (Failed status)	Individual email fails due to a problem with the address or other information.	Find out which email failed and why; look at the contents of the email table, called *DDN*_mail, which populates when the job runs. (Toad can be used for viewing DB content.) The Status column in the mail table indicates which email failed. Status can be: **Sent** – The SMTP server accepted the email for delivery. It does not necessarily mean that the user has received the email. **Unsent** – The SMTP server did not accept the email. This often means an exception was thrown. Review your error logs. **Server Error** – The SMTP server could not be contacted. The network or server may be down. **Address Error** – There is a problem with the email address on file for the recipient. The domain name may be bad (versus an invalid email format). Check the address for the failed email. If it contains a bad

Type of Failure	Probable Causes	What to Do
		domain name, correct it. **Failed** – No message from the SMTP server; may not be able to access SMTP server or other failure.
The task fails and the administrator gets no message back	Incorrect SMTP host.	Check the EmailNotification job configuration.
The task fails and generates an error or exception	Network or mail server connections not working or specifications incorrect.	Posting Service Error: PS5002 Email notification not being sent out to customers. Failed to send email. Make sure the mail server is running, check the network connectivity to the mail server, and make sure mail server specifications are correct in the EmailNotification job configuration.
The task appears to run forever.	Retry period is too large.	The default retry period is 60 minutes. Check configuration and adjust the retry period.

Using Mail Auditor

The Mail Auditor is an optional feature you can use to:

- Check the integrity of each email before the EmailNotification job sends it to the SMTP mail server.
- Maintain logs of which email was sent and which email failed the integrity checks for each EmailNotification job. Mail Auditor saves the entire text of the email and other data in logs called *.failed and *.sent in your data output directory (as you define in your Indexer job configuration).

You can use Mail Auditor with HTML or plain text mail.

The Mail Auditor performs the following checks on each email:

1. Ensures that the account number, the email address list, and the message body are not null. If the email address list is null, the account is logged to EDX_HOME/Data/ddn/*.failed with a status of “Skipped.” Here is a sample of the log format:

```
Fri May 24 18:26:55 EDT 2002 Status:Skipped DDN:NatlWireless
Account Number:0407200

Email Addresses:Message Text:

THE JOB WILL NOT FAIL FOR NULL EMAIL ADDRESS
```

To speed processing, no email message body is created and no attempt is made to send to the SMTP server for this account. No entry is made to the ddn_mail table to allow easier resending of emails by the administrator.

You can rerun the mail notification task for a particular index volume number by removing 'IVN Scanner email-Email Notification' from ddn_volume_types where z_ivn corresponds to your data file. Then rerun the EmailNotification job.

2. Verifies that there is at least one email address. See above.
3. Verifies that HTML start and end tags are correct for specific strings. If a start of email message text is provided as an environmental entry in the ejb-jar.xml, ensures that the message body starts with that text. Default is <!DOCTYPE which would be found on HTML email templates.
4. If an end of mail message text is provided as an environmental entry in the ejb-jar-.xml, ensures that the message body ends with that text. Default is </html> which would be found at the end of the HTML email template.
5. If the environmental entry for account number validation is set to true, then there should be at least one occurrence of the account number within the message body.

Mail Auditor logs the following information:

- If any of validation steps 3-5 above fail, Mail Auditor logs this information to *.failed and fails the job to alert the administrator and allow them to easily determine the problem and then retry the job. It logs the following information:

```
Mon May 06 17:23:28 EDT 2002 Status:Aborted DDN:ddn Account
Number:accountNum Email Addresses:list of emailAddresses
separated by a semicolon Message Text:Message Body
```

- If the email passes the validation steps listed above and the MailNotification job sends all email successfully, Mail Auditor logs this information to*.sent with this information:

```
Log timestamp in the format Mon May 06 17:23:28 EDT 2002
Status:Sent DDN:ddn Account Number:accountNum Email
Addresses:list of emailAddresses separated by a semicolon
Message Text:Message Body
```

- If the email passes all validation steps listed above and the email is unsuccessfully sent due to some exception, Mail Auditor logs the following information to*.failed:

```
Log timestamp in the format Mon May 06 17:23:28 EDT 2002
Status:Failed DDN:ddn Account Number:accountNum Email
Addresses:list of emailAddresses separated by a semicolon
Message Text:Message Body
```

To troubleshoot a failed EmailNotification job using Mail Auditor log files:

1. When an EmailNotification job fails, check the Data directory for the ddn and look for the *.failed files. Open this file and determine why the email failed from the logged data.
2. Correct the problem email.

3. Retry the email job in Command Center. Previously sent emails will NOT be resent.

4. Mail Auditor appends successful emails during the retry to the *.sent file.

To manage log files:

- The administrator can safely delete the sent and failed logs at any time to reclaim the drive space.

To turn Mail Auditor on/off:

You can configure each email job to use a custom Mail Auditor bean. Several beans can be in the Enterprise Archive file. All email jobs that use the same bean will use the validation parameters contained in the deployment descriptor for that bean. You can change the deployment descriptors from the WebLogic or WebSphere console.

- In the Siebel Command Center, edit the EmailNotification job configuration (MailNotification task). To turn Mail Auditor on, specify the appropriate JNDI name of your system's custom Mail Auditor bean in the Auditor Model configuration field. (For NatlWireless, use the default JNDI name edx/ejb/MailAuditor.) To turn Mail Auditor off, leave the Auditor Model field blank.

Note that running Mail Auditor slows down EmailNotification job performance.

To change the parameters Mail Auditor looks for at the beginning and end of an email:

Edit the deployment descriptor for your MailAuditor bean. The environment entries in the ejb-jar.xml looks like:

```
<env-entry>
  <description>Check For Account Number</description>
  <env-entry-name>CheckForAccountNumber</env-entry-name>
  <env-entry-type>java.lang.Boolean</env-entry-type>
  <env-entry-value>true</env-entry-value>
</env-entry>
<env-entry>
  <description>Email Message Body Start</description>
  <env-entry-name>EmailStartText</env-entry-name>
  <env-entry-type>java.lang.String</env-entry-type>
  <env-entry-value><![CDATA[<!DOCTYPE]]></env-entry-value>
</env-entry>
<env-entry>
  <description>Email Message Body end</description>
  <env-entry-name>EmailEndText</env-entry-name>
  <env-entry-type>java.lang.String</env-entry-type>
  <env-entry-value><![CDATA[</HTML>]]></env-entry-value>
</env-entry>
```

Where text in bold are the environmental entry names and cannot be changed and text in italics are the values for the environment entries that can be configured. Note that some values are enclosed in <![CDATA[…value…]]> because of the special characters like '!', '<' and '>'.

These entries are optional; if they do not occur in the deployment descriptor, then Mail Auditor does not perform the associated check.

4 DDF Validation Errors

Fixing Invalid DDF Content

If you receive errors opening or saving a DDF in DefTool or while trying to publish a DDF in Publisher, your DDF contains invalid content. DefTool also validates XML DDF content when you open or save an XML DDF. When you attempt to save a serialized DDF in Billing Manager 4.0, DefTool performs additional validations on the XML, sometimes revealing problems that previously went undetected.

DefTool displays the message “Failed during internal validation” when the schema validation fails and logs the error.

When the internal validation fails in Publisher, the message “Unable to create version set due to an invalid DDF” appears but the error is not logged.

To resolve DDF errors:

1. If the error appeared in DefTool, check your DDF error log *DDF*_error.log, where DDF is the application name, in the folder where the DDF is located for details. If you received an internal validation error in Publisher, open the DDF in DefTool and try to determine where the problem content is.
2. See the table below for resolutions to some common DDF errors. Although you can fix most problems manually, a few require you to contact Siebel Technical Support for resolution.
3. If you have saved the DDF in XML format, follow the resolution procedure for *XML* format. If you have installed 4.0 or higher but have not yet saved the serialized DDF in XML format, follow the instructions for the *serialized* DDF.

DDF Error	Resolution for XML or Serialized DDF
<*Field name*> field is not associated to a table in the DDF	*Serialized*: 1. Open the DDF in DefTool. DefTool automatically removes the field when you open the DDF. 2. Save the DDF. *XML*: 1. Open the XML DDF in an editor. 2. Search for the unassociated field name. 3. Remove all the element information. 4. Save the DDF. 5. Open the DDF in DefTool and save the file (to verify and stamp the integrity key into the DDF). Repeat this process for all unassociated elements; DefTool shows only one error at a time.
Error in reading field location for <*field name*>	*Serialized or XML*: Contact Technical Support.
Invalid page style reference <*pg style name*> for a pure/group field location Invalid page style reference <*pg style name*> for a marker location	*Serialized*: 1. In DefTool, remove the invalid page style reference from the pure/group field element or marker location and add the valid page style reference. 2. Save the DDF. *XML*: 1. Open the XML DDF using an editor. 2. Remove the invalid page style reference value from 'page-type' attribute in the corresponding field (pure/group) or marker location and add the valid page style. 3. Save the DDF. 4. Open the DDF in DefTool and save the file (to verify and stamp the integrity key into the DDF).
Invalid 'type' value for field <*field name*>	*Serialized*: 1. In DefTool, edit the field properties and add the correct field type. 2. Save the DDF. *XML*: 1. Open the XML DDF using an editor. 2. Add the correct field type value (STRING/NUMERIC/CURRENCY) in the 'data-type' attribute in the field element. 3. Save the DDF. 4. Open the DDF in DefTool and save the file (to verify and stamp the integrity key into the DDF).
Error in reading	*Serialized or XML*: Contact Technical Support.

DDF Error	Resolution for XML or Serialized DDF
locations in the marker <*marker name*>	
Contains an invalid dynamic field <*field name*> in the pattern	*Serialized*: Remove the invalid dynamic field reference: 1. In DefTool, edit the element properties and delete dynamic field reference. 2. If the 'Use Dynamic Pattern' option is set, unset it. 3. Save the DDF. *XML*: 1. Open the XML DDF in an editor. 2. Search for the element with the invalid dynamic reference. 3. Set the value of 'dynamic-pattern' attribute to "false". 4. Open the DDF in DefTool and remove the dynamic field reference as described for the serialized DDF. 5. Save the DDF.
Pattern does not contain dynamic field	*Serialized*: 1. In DefTool, edit the element properties and uncheck the 'Use Dynamic Pattern' option. 2. Save the DDF. *XML*: 1. Open the XML DDF in an editor. 2. Set the value of 'dynamic-pattern' attribute to "false" for the element. 3. Save the DDF. 4. Open the DDF in DefTool and save the file (to verify and stamp the integrity key into the DDF).
Page Style Name <*pg style name*> is invalid: Page Style, Group, Table, Marker, Column and Field names should contain alpha characters, "_" or numbers without spaces and should start with an alpha character.	*Serialized*: 1. In DefTool, remove the special characters. 2. Save the DDF. *XML*: 1. Open the XML DDF using an editor 2. Remove the special characters. 3. Save the DDF. 4. Open the DDF in DefTool and save the file (to verify and stamp the integrity key into the DDF).
Element ‘child’ is not valid for content model: ‘(table-anchor,child*)’	*Serialized or XML*: Contact Technical Support.
Error in reading location information for the table field <*tbl field name*>	*Serialized or XML:* Contact Technical Support.

DDF Error	Resolution for XML or Serialized DDF
Value <*string*> does not match regular expression facet '[^]+([ ]*[^]+)* - (schema error)	*Serialized*: 1. In DefTool, remove leading and trailing spaces of the element. 2. Save the DDF. *XML*: 1. Open the XML DDF in an editor. 2. Remove leading and trailing spaces of element. 3. Save the DDF. 4. Open the DDF in Deftool and save the file (to verify and stamp the integrity key into the DDF).
Group <*group name*> has a invalid child table <*table name*>	*Serialized*: 1. In DefTool, delete the invalid table references from the group. If you cannot see it, contact Technical Support. 2. Save the DDF. *XML*: 1. Open the XML DDF in an editor. 2. Remove the invalid child information from the group. 3. Save the DDF. 4. Open the DDF in DefTool and save the file (to verify and stamp the integrity key into the DDF).
Unable to read the DDF file	*Serialized or XML*: • Grant write permission for the DDF. If this does not work, contact Technical Support.
Multiple PRIMARY Keys are defined in the DDF	*Serialized*: 1. In DefTool, remove additional primary key fields or edit the field and unselect the field type "Primary Key" (to make it a regular field). 2. Save the DDF. *XML*: 1. Open the XML DDF using an editor. 2. Delete the additional primary key fields or change the value of 'type' attribute in field tag to "SECONDARY". 3. Save the DDF. 4. Open the DDF in DefTool and save the file (to verify and stamp the integrity key into the DDF).
Field location contains an empty pattern	*Serialized or XML:* Contact Technical Support.

5 Improving Application Performance

Summary List Response Time

The summary list, sometimes called the “hit list,” is the application view that shows indexed data for the current statement plus any historical ones available.

To determine whether your summary list response time is a problem:

1. Using a Web monitoring tool, evaluate the .jsp response times.
2. Put timestamps around the call to getSummary to see how long the call took in elapsed time.

If the hit list getSummary response time is the primary cause of delay:

1. Narrow the range of months you return, for example, display 3 months instead of 12. This reduces the number of tables that need to be searched. You could add a link if you still want users to optionally be able to access all 12 months of historical statements.
2. Improve the speed of the Index table. On DB2, run Runstats to improve the performance of database queries on the indexed data; on Oracle use Compute Statistics. You can also modify table sizes, and for Oracle, enabling native table partitioning.

Indexer Job Performance

If an application’s Indexer job is slow, here are some ways to improve performance:

- Look at the Command Center logs and determine how many minutes the Indexer and IXLoader tasks take. (Scanner and AutoIndexVolAccept tasks are not going to be the problem.) If the Indexer task is slow:
  - The Indexing DDF may not be optimized for extraction. Try making long, narrow coordinates in DefTool and simplify RegEx patterns.
  - Compression on IndexerTask might be slow.

  If the IXLoader task is slow:

- Review the load method you are using. Direct load is usually faster than Conventional load. Direct load stores data directly to the database and locks the Index table. A Conventional load uses Insert statements, one row at a time, performs multiple Selects and Inserts on the table at once, but does not lock up the table. Siebel recommends using Direct load for Oracle and SQL Server, and Conventional load for DB2. See the *Siebel Billing Manager Administration Guide* for details on the IXLoader task configuration options.

- There is an approximate 20-second overhead for processing individual data files. Using many small files can significantly impact the Indexer job. If possible, use fewer input files.

EmailNotification Job Performance

The Siebel email solution platform can scale to move more than one million emails/day. We are capable of 100,000/hr. If you get drastically slower performance, we have found that the majority of problems can be fixed by plug-in or by optimizing the SMTP server.

If an application's EmailNotification job is slow, here are some ways to improve performance:

- Review the response time of your email account resolver. Check how long it takes per lookup. Add elapsed time logging to your plug in so you know the moment it came in to the moment it returned the email address (the start and end of the method).

- Your SMTP server may be slow. To test, add multiple IP addresses for SMTP servers, comma-separated, to the "SMTP Hosts" field of your EmailNotification job configuration. This randomly selects SMTP servers to send email to. If the problem was your SMTP server, you should see a dramatic improvement in response time. Optimize or tune up the SMTP server if it is slow.