CHAPTER 5

Troubleshooting CDS Video Navigator and Poster Art Server

This chapter contains information that is useful for identifying and remedying problems related to Cisco CDS Video Navigator and Poster Art Server. This chapter presents the following major topics:

Troubleshooting Video Navigator

The following tasks are presented:

Using Video Navigator Logging and Log Files

Video Navigator log file entries can provide useful information for troubleshooting. Video Navigator uses the Apache Tomcat application server, which uses the log4j logging system.

The Video Navigator log file is named midas.log and resides in the /home/isa/MIDAS/logs directory. The log file is automatically rotated when it reaches 20 MB. Up to ten archived log files are stored in the /home/isa/MIDAS/logs directory, as in the following example:

```
-rw-rw-r-- 1 isa isa 342473 Jan 23 20:40 midas.log
-rw-rw-r-- 1 isa isa 20965936 Jan 19 17:45 midas.log.1
-rw-rw-r-- 1 isa isa 20960394 Jan 18 14:00 midas.log.2
-rw-rw-r-- 1 isa isa 20966090 Jan 17 20:52 midas.log.3
-rw-rw-r-- 1 isa isa 20968401 Jan 16 20:52 midas.log.4
-rw-rw-r-- 1 isa isa 20969543 Jan 15 20:51 midas.log.5
-rw-rw-r-- 1 isa isa 20969613 Jan 14 20:51 midas.log.6
-rw-rw-r-- 1 isa isa 20968092 Jan 13 20:51 midas.log.7
-rw-rw-r-- 1 isa isa 20966422 Jan 12 20:51 midas.log.8
-rw-rw-r-- 1 isa isa 20969681 Jan 11 20:51 midas.log.9
```

Logging Levels

The Video Navigator logging levels are FATAL, ERROR, WARN, INFO, and DEBUG. Logging levels go from least verbose to most verbose. The FATAL level generates the smallest number of messages, and the DEBUG level generates the greatest number of messages. The default logging level is WARN.

In a lab environment, consider keeping the logging level at DEBUG. However, using the DEBUG level affects performance. For load testing or a production environment, the logging level should be set to INFO or, for better performance, to WARN.

The Video Navigator logging configuration file is named midasLog.properties and resides in the /home/isa/MIDAS/config directory.

Do the following to change the logging level:

Step 1 If needed, log in as root on the CDE110 that hosts Video Navigator.

Step 2 Change the working directory as follows:

```
[root@system]# cd /home/isa/MIDAS/config
```

Step 3 Use a text editor to modify the midasLog.properties file.

a. Change the logging level in the property log4j.logger.Midas Logger to the applicable value. Allowed values are FATAL, ERROR, WARN, INFO, and DEBUG. For example, the following line changes the log level to DEBUG:

```
log4j.logger.MidasLogger=DEBUG, MidasLogFile
```

b. Save and close the midasLog.properties file.

Note Changes to the midasLog.properties file do not take effect until Video Navigator is restarted in Step 4.

Step 4 To stop and restart Video Navigator, issue the following commands:

```
[root@system]# stop_midas

MIDAS Running ......shutting down
Using CATALINA_BASE: /usr/local/tomcat
Using CATALINA_HOME: /usr/local/tomcat
Using CATALINA_TMPDIR: /usr/local/tomcat/temp
Using JRE_HOME: /usr/local/java
Killing: 17895
MIDAS stopped

[root@system]# start_midas

MIDAS not running ....... starting MIDAS
```

Log Entry Format

Each Video Navigator log entry has the following format:

[*Date Time*] [*Logging level*] [*Logging class*] - [*Log message*]

Some sample log entries are as follows:

```
[2008-02-04 18:49:49,626][ERROR] [CatalogHttpServer] - Error fetching catalog from
Tandberg java.io.IOException: Method failed: HTTP/1.1 503 Service Temporarily Unavailable

[2008-02-04 18:45:29,322][WARN] [BrowseCatalogAction] - device: 132 - Invalid page size -1
[2008-02-04 18:45:58,290][WARN] [BrowseCatalogAction] - device: 132 - Invalid page size -1
[2008-02-04 18:46:04,971][WARN] [BrowseCatalogAction] - device: 132 - Catalog 100 doesn't
exist

[2008-02-04 18:49:49,627][INFO] [CatalogHttpServer] - Fetch local catalog
[2008-02-04 18:49:49,825][INFO] [CatalogHttpServer] - Last local catalog
[2008-02-04 18:49:49,826][INFO] [CatalogHttpServer] - CR catalogId = 1
[2008-02-04 18:49:49,827][INFO] [CatalogHttpServer] - CR downloadTime = 1201219710722
[2008-02-04 18:49:49,828][INFO] [CatalogHttpServer] - CR fileName = catalog.113.xml
[2008-02-04 18:49:49,828][INFO] [CatalogHttpServer] - Going to load local catalog file
/home/isa/MIDAS/catalog/catalog.113.xml

[2008-02-04 18:49:50,133][DEBUG] [AssetIndexer] - Built index for asset 327
[2008-02-04 18:49:50,133][DEBUG] [AssetIndexer] - Add title = Wedding Crashers
[2008-02-04 18:49:50,134][DEBUG] [AssetIndexer] - Add summary = Two divorce mediators who
crash weddings to pick up women engage in comical misadventures at the nuptials of a
politician's daughter. Ends 09/02
[2008-02-04 18:49:50,134][DEBUG] [AssetIndexer] - Add content = Wedding Crashers Two
divorce mediators who crash weddings to pick up women engage in comical misadventures at
the nuptials of a politician's daughter. Ends 09/02 Owen Wilson Vince Vaughn Christopher
Walken Rachel McAdams
[2008-02-04 18:49:50,142][DEBUG] [AssetIndexer] - Built index for asset 328
[2008-02-04 18:49:50,142][DEBUG] [AssetIndexer] - Add title = The Rock
[2008-02-04 18:49:50,143][DEBUG] [AssetIndexer] - Add summary = Sean Connery and Nicolas
Cage team up to stop a devious U.S. general's threat to chemically attack San Francisco
from Alcatraz Island. Ends 09/09
[2008-02-04 18:49:50,160][DEBUG] [AssetIndexer] - Add content = The Rock Sean Connery and
Nicolas Cage team up to stop a devious U.S. general's threat to chemically attack San
Francisco from Alcatraz Island. Ends 09/09 Sean Connery Nicolas Cage Ed Harris Michael
Biehn William Forsythe
```

Understanding the Video Navigator Directory Structure

Video Navigator uses the directory structure shown in Table 5-1.

Table 5-1 Video Navigator Directory Structure

Directory	Contents
/home/isa/MIDAS/catalog	Latest catalog files fetched from the VBO. A maximum of five latest catalog files are kept in this directory. ⚠ **Caution** *Do not modify the contents of this directory.*
/home/isa/MIDAS/config	Files for configuring Video Navigator
/home/isa/MIDAS/docs	Release notes or some user help files
/home/isa/MIDAS/epg/data	EPG data and a mapping file
/home/isa/MIDAS/epg/index	EPG search indexes
/home/isa/MIDAS/logs	Video Navigator log files. For more information, see the "Using Video Navigator Logging and Log Files" section on page 5-2.
/home/isa/MIDAS/scripts	Scripts for starting, stopping, and checking whether Video Navigator is running
/home/isa/MIDAS/test	Sample test files
/home/isa/MIDAS/webapps	Video Navigator web applications.
/usr/local/apache2	Apache web server binary executable
/usr/local/tomcat	Files related to the Tomcat application server

Using Video Navigator Scripts

The Video Navigator software includes the scripts shown in Table 5-2. These scripts may be needed to troubleshoot and resolve some Video Navigator problems.

Table 5-2 Video Navigator Scripts

Directory and Script	Contents
/home/isa/MIDAS/scripts/start_midas	Starts Video Navigator and the Apache Tomcat application server (tomcat5).
/home/isa/MIDAS/scripts/stop_midas	Stops Video Navigator.
/home/isa/MIDAS/scripts/check_midas	Checks whether Video Navigator is running and displays the release number of the installed Video Navigator software.
/home/isa/MIDAS_IntegrationTest/clientinterfacetest	Checks whether the client-facing web services interface is working correctly. The **clientinterfacetest** command does not verify the connection from Video Navigator to the STB.
/home/isa/MIDAS_IntegrationTest/searchTest	Checks whether EPG and VoD search are working.

Note The Video Navigator software installation script adds /home/isa/MIDAS/scripts to the PATH variable but does not add /home/isa/MIDAS_IntegrationTest to the PATH. Therefore, your current working directory must be /home/isa/MIDAS_IntegrationTest when the clientinterfacetest script is executed.

Troubleshooting Specific Problems in Video Navigator

This section provides information on troubleshooting.

Problems with VOD Catalog Not Displaying on the STB

If the VOD catalog is not available on an STB, set the Video Navigator logging level to DEBUG. For information on how to set the logging level, see the "Logging Levels" section on page 5-2.

Observe the entries in the log file for SignOn and BrowseCatalog requests. When the STB user selects the VOD service, the Video Navigator client on the STB makes a SignOn request to Video Navigator. The Video Navigator client queries Video Navigator by device ID (MAC address of the STB) for the list of VoD services to which it is entitled. With the logging level set to DEBUG, you should be able to see the SignOn request in the Video Navigator log file, as in the following example:

```
[2008-01-31 11:10:38,125][DEBUG] [SignOnAction] - SignOn request
[2008-01-31 11:10:38,125][DEBUG] [SignOnAction] - deviceId = null
[2008-01-31 11:10:38,128][DEBUG] [SignOnAction] - deviceId 00:14:F8:E3:0A:DF- SignOn:
processServices
[2008-01-31 11:10:38,152][DEBUG] [SignOnAction] - Process service done
[2008-01-31 11:10:38,152][DEBUG] [SignOnAction] - deviceId 00:14:F8:E3:0A:DF- SignOn:
processRentals
[2008-01-31 11:10:38,221][DEBUG] [SignOnAction] - deviceId 00:14:F8:E3:0A:DF has 0 rental
records on Tandberg system
[2008-01-31 11:10:38,221][DEBUG] [SignOnAction] - No of rental records in MIDAS database =
0
[2008-01-31 11:10:38,222][DEBUG] [SignOnAction] - Process rental done
[2008-01-31 11:10:38,222][DEBUG] [ActionSkeleton] - HTTP Response
[2008-01-31 11:10:38,222][DEBUG] [ActionSkeleton] - <?xml version="1.0" encoding="UTF-8"?>
```

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"><env:Body
xmlns="com.cisco.midas.client"><SignOnResponse /></env:Body></env:Envelope>^
```

When the STB user starts browsing the VOD catalog, the Video Navigator client on the STB makes a BrowseCatalog request to Video Navigator. The following log entries show a BrowseCatalog request-and-response example:

```
[2008-01-31 11:10:38,334][DEBUG] [BrowseCatalogAction] - BrowseCatalog request
[2008-01-31 11:10:38,335][DEBUG] [BrowseCatalogAction] - deviceId = 00:14:F8:E3:0A:DF
[2008-01-31 11:10:38,335][DEBUG] [BrowseCatalogAction] - catalogId = 1
[2008-01-31 11:10:38,335][DEBUG] [BrowseCatalogAction] - categoryId = 0
[2008-01-31 11:10:38,338][DEBUG] [BrowseCatalogAction] - pageNo = 1
[2008-01-31 11:10:38,338][DEBUG] [BrowseCatalogAction] - pageSize = 7
[2008-01-31 11:10:38,338][DEBUG] [BrowseCatalogAction] - Construct JDOM response
[2008-01-31 11:10:38,339][DEBUG] [BrowseCatalogAction] - deviceId 00:14:F8:E3:0A:DF - From
= 0 to = 5
[2008-01-31 11:10:38,339][DEBUG] [BrowseCatalogAction] - deviceId 00:14:F8:E3:0A:DF -
Output list size = 5
[2008-01-31 11:10:38,339][DEBUG] [BrowseCatalogAction] - hasCategory
[2008-01-31 11:10:38,339][DEBUG] [BrowseCatalogAction] - hasServiceCategory
[2008-01-31 11:10:38,339][DEBUG] [ActionSkeleton] - HTTP Response
[2008-01-31 11:10:38,340][DEBUG] [ActionSkeleton] - <?xml version="1.0"
encoding="UTF-8"?>^M
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"><env:Body
xmlns="com.cisco.midas.client"><BrowseCatalogResponse><category id="2" name="All Movies"
parentId="0" desc="" /><category id="4" name="Cat1" parentId="0" desc="" /><category
id="12" name="HBO" parentId="0" desc="" /><serviceCategory id="15" name="MAX" parentId="0"
desc=""><service id="203" name="MAX" desc="Cinemax On Demand" price="$2.99"
purchased="false"><backoffice name="vendorName" value="Tandberg" /><backoffice
name="vendorVersion" value="3.3" /><backoffice name="serviceOfferingId" value="203"
/><backoffice name="serviceName" value="MAX"
/></service></serviceCategory><serviceCategory id="20" name="Showtime" parentId="0"
desc=""><service id="255" name="SOD" desc="Showtime On Demand" price="$5.99"
purchased="false"><backoffice name="vendorName" value="Tandberg" /><backoffice
name="vendorVersion" value="3.3" /><backoffice name="serviceOfferingId" value="255"
/><backoffice name="serviceName" value="SOD"
/></service></serviceCategory><pageNo>1</pageNo><pageSize>7</pageSize><totalItems>5</total
Items><hasMore>false</hasMore><catalogVersion>1</catalogVersion></BrowseCatalogResponse></
env:Body></env:Envelope>
```

It is possible that the SignOn and BrowseCatalog requests and responses appear in the Video Navigator log entries, but no VoD catalog is displayed on the STB. This set of symptoms may indicate that there is a network problem or (less likely) that the STB cannot process the returned data.

If there is no log-file entry for the SignOn request or other activity in the Video Navigator log file, follow these steps:

Step 1 Log in as root.

Step 2 If you know the IP address of the STB, use the **ping** command and the IP address to verify that the Video Navigator server (CDE110) can reach the STB.

Step 3 To check that Video Navigator is running, issue the following command:

```
[root@system]# check_midas
```

If Video Navigator is running, “MIDAS is running” is returned. If Video Navigator is not running, “MIDAS is not running” is returned.

Do one of the following:

- If Video Navigator is not running, go to Step 5.

- If Video Navigator is running, go to Step 4.

Step 4 If Video Navigatorr is running, verify that the Apache HTTP server service (httpd) is running by issuing the following command:

```
[root@system]# ps -ef | grep httpd
```

Do one of the following depending on whether httpd is running:

- If httpd is running, the output is similar to the following. Go to Step 6.

```
root      2880     1  0 Jul18 ?        00:00:00 /usr/sbin/httpd
apache    4881  2880  0 04:03 ?        00:00:00 /usr/sbin/httpd
apache    4882  2880  0 04:03 ?        00:00:00 /usr/sbin/httpd
apache    4883  2880  0 04:03 ?        00:00:00 /usr/sbin/httpd
apache    4884  2880  0 04:03 ?        00:00:00 /usr/sbin/httpd
apache    4885  2880  0 04:03 ?        00:00:00 /usr/sbin/httpd
apache    4886  2880  0 04:03 ?        00:00:00 /usr/sbin/httpd
apache    4887  2880  0 04:03 ?        00:00:00 /usr/sbin/httpd
apache    4888  2880  0 04:03 ?        00:00:00 /usr/sbin/httpd
```

- If httpd is not running, the preceding output will not be present. Restart the httpd daemon by issuing the following command:

```
[root@system]# service httpd restart
```

Go to Step 6.

Step 5 If Video Navigator is not running, restart it and verify that it is running by issuing the following commands:

```
[root@system]# start_midas

MIDAS not running ....... starting MIDAS

[root@system]# check_midas
```

Note Because CD-VN is a web application, there is no Video Navigator process that you can check to see if it is running. Instead, use the **check_midas** command.

```
MIDAS (10.1.X.X) is running
```

Step 6 On the STB, select or have the user select the VoD service.

Step 7 Check the log entries to verify that a SignOn request from the STB is sent to Video Navigator.

Troubleshooting Poster Art Server

This section covers issues that are specific to Poster Art Server.

The following tasks are presented:

- Using Poster Art Server Logging and Log Files
- Understanding the Poster Art Server Directory Structure
- Troubleshooting Poster Art Server Sequence Flows

Using Poster Art Server Logging and Log Files

After Poster Art Server is installed, it uses the default Apache log4j configuration file, log4j.properties:

`<webapps_path>/PAServer/WEB-INF/classes/log4j.properties`

Table 5-3 lists the components of the log4j.properties file.

Table 5-3 Components of log4j.properties

Component	Description/Example
log4j.appender.PasLogFile	org.apache.log4j.RollingFileAppender
log4j.appender.PasLogFile.File	/home/isa/MIDAS/logs/PosterServer.log
log4j.appender.PasLogFile.MaxFileSize	40 MB
log4j.appender.PasLogFile.MaxBackupIndex	5
log4j.appender.PasLogFile.layout	org.apache.log4j.PatternLayout
log4j.appender.PasLogFile.layout.ConversionPattern	[%d][%p] [%C{1}] - %m%n
log4j.logger.com.cisco	debug, PasLogFile

Each log message is in the following format:

<local time>: <severity>: <operation>: <error code>: <error message string>

The following are example log messages:

```
[2010-02-25 23:50:32,110][DEBUG] [PAServerInit] - Complete to init DBConnection
[2010-02-25 23:50:32,110][DEBUG] [PAServerInit] - Init task manager
[2010-02-25 23:50:32,118][DEBUG] [PAServerInit] - Init image store
[2010-02-25 23:50:32,123][DEBUG] [PAServerInit] - Init CSI interface
```

Understanding the Poster Art Server Directory Structure

Poster Art Server uses the directory structure shown in Table 5-4.

Table 5-4 Poster Art Server Directory Structure

Directory	Description/Contents
/home/isa/MIDAS/webapps/PAServer	Root of Poster Art Server application
/home/isa/MIDAS/webapps/PAServer/WEB-INF/web.xml	Poster Art Server configuration files
/home/isa/MIDAS/config/paslog.properties	Poster Art Server log configuration file. If this file does not exist, the log file is /home/isa/MIDAS/webapps/PAServer/WEB-INF/classes/log4j.properties

Troubleshooting Poster Art Server Sequence Flows

This section presents some detailed sequence flows and explains how to troubleshoot them.The sequence flow for each use case lists a set of error conditions that are logged to the application log file, pas.log. (See Using Poster Art Server Logging and Log Files, page 5-8).

Note Syslog logging is not supported in this release of Poster Art Server.

Table 5-5 describes the Poster Art Server log message formats and provides examples.

Table 5-5 Poster Art Server Log Message Formats

Element	Description	Value	Example
local time	Local time code	Any ASCII string	Tue Dec 23 02:30:06 2008
Error code	Unique code for each error See Poster Art Server Error Messages, page 5-17.	1xx: Internal System Error	100
		2xx: Network Communication Error	200
		3xx: Communication Message Error	300
		Fatal	Warn

Table 5-5 Poster Art Server Log Message Formats (continued)

Element	Description	Value	Example
Severity	Degree of impact to the system	Fatal	Warn
		Error	
		Warn	
		Info	
		Debug	
		Trace	
Operation	Brief description of the operation being attempted when the error occurred	SystemStartup	
		DBconnect	DBconnect
		GetAvailableSpace	
		GetAssetCount	
		GetAssetsInfo	
		GetAssetsStatus	
		GetAssetInventoryList	
		AddAsset	
		UpdateAsset	
		DeleteAsset	
		GetPoster	

The following is an example log message:

```
Tue Dec 23 02:30:06 2008 : Critical : 300 : SystemStartup : Could not find configuration
file config.xml
```

Note The format of the above message illustrates the default. The operator can customize this format by modifying pasLog.properties.

For the list of all error codes and error strings, see the "Poster Art Server Error Messages" section on page 5-17.

Sequence Flows

This section presents a variety of practical sequence flows that can be used in troubleshooting:

- SystemStartup
- GetAvailableSpace
- GetAssetInventoryList
- AddAsset
- UpdateAsset
- DeleteAsset
- GetPoster
- SystemStop

SystemStartup

The following components are involved in this sequence:

VBO <> CSIService <> ImageStore <> Database

1. ImageStore sets up a persistent connection with the database.
2. CSIService sends a SignOn request to the VBO.
3. The VBO responds to the message. If the response message reports an error, CSIService keeps retrying until it gets a successful response.
4. CSIService starts listening at the configured port.

Note To synchronize with the poster data, and ensure system resilience, the VBO must execute a sign-on. For more information, see the "SignOn" section on page 4-25.

The system does not load poster information from the database at start up, but rather waits until it receives a sign-on request.

Table 5-6 lists and describes the error codes for SystemStartup.

Table 5-6 Error Codes and Descriptions for SystemStartup

Error code	Description
100	Load configuration file error. Poster Art Server uses the default configuration, and logs a warning message in the log file.
101	Read configuration parameter error. Poster Art Server uses the default parameter, and logs a warning message in the log file.
102	ImageStore connection to database error. Poster Art Server keeps retrying, and logs a fatal error message in log file.
103	CSIService fails to listen at the configured port. Poster Art Server quits, and logs a fatal error message in the log file. Troubleshoot to resolve the problem.
201	VBO server is down, and sign on fails. Poster Art Server keeps retrying, and logs a fatal error message in the log file.
202	VBO server responds that sign-on has failed. Poster Art Server keeps retrying, and logs a fatal error message in the log file.

GetAvailableSpace

The following components are involved in this sequence:

VBO <> CSIService <> ImageStore <> FileSystem

1. The VBO sends a GetAvailableSpace request to the listening address. CSIService gets the request and parses it.
2. CSIService calls the ImageStore by using the GetAvailableSpace method to get the size of the available space.
3. ImageStore makes a request to the native file system to get the size of the available space.
4. CSIService constructs an HTTP response message.
5. CSIService sends the response message back to the VBO.

Table 5-7 lists and describes the error codes for GetAvailableSpace.

Table 5-7 Error Codes and Descriptios for GetAvailableSpace

Error code	Description
301	Invalid asset type in request. Poster Art Server responds with an InvalidAssetException to the VBO and logs the error.
104	Native file system returns an error message when ImageStore issues a GetAvailableSpace request. Poster Art Server responds to the VBO with an InternalErrorException and logs the error.

GetAssetInventoryList

The following components are involved in this sequence:

VBO <> CSIService <> ImageStore <> Database

1. The VBO sends a GetAssetInventoryList request to the listening address. CSIService gets the request and parses it.
2. CSIService calls the ImageStore by using the GetAssetList method to get all asset information.
3. ImageStore sends a GetAll request to the database.
4. The database returns the asset information list.
5. ImageStore passes the asset information list to CSIService.
6. CSIService constructs an HTTP response message.
7. CSIService sends the response message back to the VBO.

Table 5-8 lists and describes the error codes for GetAssetInventoryList.

Table 5-8 Error Codes and Descriptions for GetAssetInventoryList

Error code	Description
301	Invalid asset type in request. Poster Art Server responds to the VBO with an InvalidAssetException and logs the error.
105	Database transaction error. Poster Art Server responds to the VBO with an InternalErrorException and logs the error.

AddAsset

The following components are involved in this sequence:

VBO <> CSIService <> TaskManager <> Task <> ImageStore <> Connection <> ImageProcessor <> Database <> VBO Storage

1. The VBO sends an AddAsset request to the listening address. CSIService gets the request and parses it.
2. CSIService adds the request to the TaskManager asynchronous task queue, and a check is made to see if the asset is already in the system.
3. TaskManager creates the poster information entry in the database, and sets the state to Pending.
4. CSIService constructs an HTTP response message and sends it to the VBO.
5. When it is time for TaskManager to process a new request, it checks out a new task from the Task Queue.
6. If TaskManager needs a task, it creates one. For the algorithm used, see the "GetAssetInventoryList" section on page 5-13.
7. Task creates a Connection object.
8. TaskManager starts the task to download the poster.
9. Task sets the state of the poster to Transferring.
10. Task asks the Connection object to connect to the target server.
11. Connection makes a connection to the target server.

12. Task calls the AddImage method of ImageStore to add the poster file.
13. ImageStore calls the Connection object to start the download.
14. The Connection object communicates with the target server to download the data.
15. After the download is complete, ImageStore processes the image.
16. ImageStore works with ImageProcessor to implement some image processing features, such as resizing.
17. ImageStore stores the poster file in the corresponding local directory.
18. Task asks Connection to disconnect from the target server.
19. Connection disconnects from the target server.
20. Task sets the state of the poster to Completed if the operation is successful, or to Failed if a failure occurs in any of the preceding steps.
21. Task constructs a callback message and sends it to the VBO.
22. Task communicates with TaskManager to check in and terminate itself.

Table 5-9 lists error codes and descriptions for AddAsset.

Table 5-9 Error Codes and Descriptions for AddAsset

Error code	Description
301	Invalid asset type in the request. Poster Art Server responds to the VBO with InvalidAssetType and logs the error.
302	Asset exists in the system with Completed status and the same version that the VBO has for it. Poster Art Server responds to the VBO with VersionAlreadyExists and logs the error.
303	Transport protocol in the request message has the wrong format. Poster Art Server responds to the VBO with InvalidTransferProtocol, and logs the error.
304	Callback address in the request message is in the wrong format. Poster Art Server responds to the VBO with InvalidCallbackAddress, and logs the error.
105	Database transaction error. Poster Art Server responds to the VBO with InternalErrorException and logs the error. The state of the poster is set to Failed and the VBO must ingest the poster again.
203	Problem with VBO repository server. Poster Art Server sends VBORepositoryErrorException to the VBO in the callback message and logs the error. The state of the poster is set to Failed and the VBO must ingest the poster again.
106	Problem with resizing. Poster Art Server responds to the VBO with InternalErrorException and logs the error. The state of the poster is set to Failed and the VBO must ingest the poster again.
107	File system error. Poster Art Server responds to the VBO with InternalErrorException, and logs the error. The state of the poster is set to Failed and the VBO must ingest the poster again.

Table 5-9 Error Codes and Descriptions for AddAsset (continued)

Error code	Description
305	Asset in the AddAsset request exists and is in Transferring state. Poster Art Server responds to the VBO with InvalidId exception.
204	Callback server is unreachable after maximum number of retries. Poster Art Server only logs the error, because there is no way to contact the VBO.
109	Available space is used up. Poster Art Server responds to the VBO with InternalErrorException, and logs the error. The state of the poster is Pending and the VBO requires troubleshooting.

UpdateAsset

The following components are involved in this sequence:

VBO <> CSIService <> TaskManager <> ImageStore <> Connection <> ImageProcessor <> Database
<> VBO Storage

This is similar to the "AddAsset" section on page 5-13, except that the following steps are inserted between Step 12 and Step 13:

1. Call the UpdateImage method of an ImageStore instance, instead of calling AddImage.
2. ImageStore gets the poster information from the database.
   a. If the poster version is different from the existing version, Poster Art Server deletes the old one, including resized poster files, from ImageStore.
   b. If the poster version is the same as the requested one, Poster Art Server responds with a VersionAlreadyExists exception.

Error codes are identical to those in Table 5-9 on page 5-14, except that AddAsset is replaced with UpdateAsset.

DeleteAsset

The following components are involved in this sequence:

VBO <> CSIService <> TaskManager <> Task <> ImageStore <> Database

1. The VBO sends a DeleteAsset request to the listening address. CSIService gets the request and parses it.
2. CSIService adds the request into the TaskManager asynchronous task queue.
3. CSIService constructs an HTTP response message and sends it to the VBO.
4. When it is time for TaskManager to process a new request, it checks out a new task from the task queue. Any available task can process this job.
5. TaskManager places the callback information into the Task object.
6. If the state of the asset is not Transferring, TaskManager starts the task to delete the poster.
7. Task calls the DeleteImage method of ImageStore to delete the poster file.
8. ImageStore deletes the poster file, including resized ones, from the corresponding local directory.
9. ImageStore deletes the poster information from the database.

10. Task constructs a callback message and sends it back to the VBO.
11. Task checks in itself into TaskManager and terminates.

Table 5-10 lists error codes and descriptions for DeleteAsset.

Table 5-10 Error Codes and Descriptions for DeleteAsset

Error code	Description
301	Invalided asset type in request. Poster Art Server responds to the VBO with InvalidAssetException, and logs the error.
306	Version requested is different from the one in the system. Poster Art Server responds to the VBO with VersionDoesNotExist, and logs the error.
105	Database transaction error. Poster Art Server responds to the VBO with InternalErrorException, and logs the error. The VBO must delete the poster again.
107	File system error. Poster Art Server responds to the VBO with InternalErrorException, and logs the error. The VBO must delete the poster again.

GetPoster

The following components are involved in this sequence:

STB <> PosterHttpServer <> ClientService <> ImageStore <> Image

1. The STB sends a GetPoster request to the Poster Art Server.
2. PosterHTTPServer gets and parses the request, then calls the ClientService GetPoster method.
3. ClientService calls the GetImage method of ImageStore, with the input arguments of xdim and ydim. ImageStore creates an Image object, assigns the correct local path to it, and returns the object to the STB.
4. ClientService calls the GetImageData method of the Image object to get binary data.
5. CSIService constructs a response message and returns it to PostHTTPServer.
6. PostHTTPServer constructs an HTTP response message and sends it back to the STB.

Table 5-11 lists error codes and descriptions for GetPoster.

Table 5-11 Error Codes and Descriptions for GetPoster

Error code	Description
105	Database transaction error. Poster Art Server responds to the the STB with a MIDASInternalSystemError exception and logs the error. An exception is returned either as a SOAP message (as a soapenv:Fault element within the SOAP envelope), or as just a plain XML message.

Table 5-11 Error Codes and Descriptions for GetPoster (continued)

Error code	Description
107	File system error. Poster Art Server responds to the the STB with a MIDASInternalSystemError exception and logs the error.
108	Poster does not exist. Poster Art Server responds to the the STB with a MIDASInvalidPosterId exception and logs the error.

SystemStop

When Poster Art Server service is stopped, CSIService sends a SignOff request to the VBO. However, if any error occurs it does not retry, and a log is written in the log file.

Table 5-12 lists error codes and descriptions for SystemStop.

Table 5-12 Error Codes and Descriptions for SystemStop

Error code	Description
201	VBO server is down, sign off failed. Poster Art Server logs a fatal error message in the log file.
202	VBO server response sign on failed. Poster Art Server logs a fatal error message in the log file.

Poster Art Server Error Messages

Table 5-13 lists all of the error codes and strings that are used for Poster Art Server.

Table 5-13 Poster Art Server Error Codes and Strings

Error code	Error string
100	The system was unable to load the configuration file (By default, the configuration file name is web.xml. The log property file name is pasLog.properties.)
101	Failed to read configuration parameter
102	Connections database error
103	Failed to listen at configured port
104	Get disk info error
105	Database transaction error
106	Failed to resize
107	IO error
108	Poster does not exist
109	Available space is used up
201	Failed to connect remote server
202	Remote server sign on failure
203	VBO repository server has problem

Table 5-13 Poster Art Server Error Codes and Strings (continued)

Error code	Error string
204	Call back server has problem
301	Invalid asset type
302	Asset version exists in system
303	Invalid transport protocol
304	Invalid callback address
305	Invalid asset ID
306	The asset version in request is different from the one in system
307	Wrong format or version
308	Asset is in processing progress

web.xml

The following is an example of the configuration file web.xml.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app id="WebApp_ID" version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
    <display-name>PosterArtServer</display-name>

    <servlet>
        <servlet-name>PAServerInit</servlet-name>
        <servlet-class>com.cisco.paserver.PAServerInit</servlet-class>

        <init-param>
           <param-name>log4j-init-file</param-name>
           <param-value>/home/isa/MIDAS/config/paslog.properties</param-value>
        </init-param>

        <!-- If set SignOnOffMode to be 1, poster art server will sign on vbo -->
        <init-param>
          <param-name>SignOnOffMode</param-name>
          <param-value>0</param-value>
        </init-param>

    <!-- PAServerIp, PAServerPort and PAServerName are used to construct vbo sign-on
request-->
        <init-param>
           <param-name>PAServerIp</param-name>
           <param-value>127.0.0.1</param-value>
        </init-param>

        <init-param>
           <param-name>PAServerPort</param-name>
           <param-value>8088</param-value>
        </init-param>

        <load-on-startup>1</load-on-startup>
    </servlet>

    <servlet>
```

```
    <description>
    </description>
    <display-name> CSIHttpServer</display-name>
    <servlet-name>CSIHttpServer</servlet-name>
    <servlet-class> com.cisco.paserver.csi.CSIHttpServer</servlet-class>
</servlet>
<servlet>
    <description>MIDAS Poster Server</description>
    <display-name>PosterHttpServer</display-name>
    <servlet-name>PosterHttpServer</servlet-name>
    <servlet-class>com.cisco.midas.client.PosterServer</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>CSIHttpServer</servlet-name>
    <url-pattern>/CSI/*</url-pattern>
</servlet-mapping>
<servlet-mapping>
    <servlet-name>PosterHttpServer</servlet-name>
    <url-pattern>/PosterHttpServer/*</url-pattern>
</servlet-mapping>

<welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
</welcome-file-list>

<context-param>
    <param-name>MaxRetry</param-name>
    <param-value>3</param-value>
</context-param>

<context-param>
    <param-name>RetryInterval</param-name>
    <param-value>3</param-value>
</context-param>

<context-param>
    <param-name>TimeOut</param-name>
    <param-value>3</param-value>
</context-param>

<context-param>
    <param-name>ResizeOption</param-name>
    <param-value>352x288:480x480:600x600</param-value>
</context-param>

<context-param>
    <param-name>DBHost</param-name>
    <param-value>127.0.0.1</param-value>
</context-param>

<context-param>
    <param-name>DBPort</param-name>
    <param-value>9999</param-value>
</context-param>

<context-param>
    <param-name>MinDBConnection</param-name>
    <param-value>3</param-value>
</context-param>
```

```
    <context-param>
        <param-name>MaxDBConnection</param-name>
        <param-value>8</param-value>
    </context-param>

    <context-param>
        <param-name>ImageStoreRootPath</param-name>
        <param-value>/tmp/pasroot</param-value>
    </context-param>

    <context-param>
        <param-name>VboUrl</param-name>
        <param-value>http://127.0.0.1:8000/sign</param-value>
    </context-param>

</web-app>
```