CHAPTER 3

Understanding and Using the Cisco CDS Poster Art Server

This chapter discusses the Cisco Content Delivery System (CDS) Poster Art Server and contains the following major topics:

For a high-level overview, see "Cisco CDS Poster Art Server Overview" section on page 1-4.

Basic Operation

The basic operation of Poster Art Server is as follows:

1. The poster art client on the STB sends an HTTP GET request to the PAS, in the following example format:

   `http://<host name>:<port>/PAServer/PosterHttpServer/GetPoster?id=<poster id>&xdim=<xdim>&ydim=<ydim>`

   where

   - <*host name*> naturally varies
   - <*port*> is optional, and depends on the type of web server that is deployed (for example, Apache, Apache Tomcat)
   - <*poster id*> is the ID of the poster that is defined and established in the CSI request SetAssetRequest for poster ingestion (see Chapter 4, "Using the Cisco Content Storage Interface")
   - <*xdim*> and <*ydim*> are optional parameters (see the "Storing and Resizing Images" section on page 3-9)

   The following is an example request without dimensions specified:

   `http://10.66.4.14/PAServer/PosterHttpServer/GetPoster?id=2babec60-31db-46b8-80a7-231a08ce64ee`

   The following is an example request with dimensions specified:

```
http://10.66.4.14/PAServer/PosterHttpServer/GetPoster?id=2babec60-31db-46b8-80a7-231a0
8ce64ee&xdim=80&ydim=120
```

2. Poster Art Server delivers the image file to the client.

Note Because the VBO synchronizes all poster data, it is possible for the STB to require a poster that is already in the asset repository, but Poster Art Server has not yet downloaded the entire file. As a result, the STB receives incorrect poster data. In this case, Poster Art Server responds with an InvalidPosterId exception. The STB then tries to download another poster.

Poster Art Server Components and Functions

This section discusses the following components and functions:

- Poster Art Server Logical Interfaces
- Role of the TaskManager Module
- Poster States

Poster Art Server Logical Interfaces

Poster Art Server is located between the VBO and the STB client. All the interfaces for Poster Art Server use XML over HTTP. Poster Art Server communicates with external components through two interfaces:

- Cisco Content Storage Interface (CSI)—CSI supports interactions for ingesting and managing poster art on Poster Art Server.
- Cisco Client API (Video Navigator client-facing API)—The client uses this interface to fetch the poster art that is needed.

For details about Cisco CSI, see Chapter 4, "Using the Cisco Content Storage Interface."

Poster Art Server uses the software modules listed in Table 3-1.

Table 3-1 Poster Art Server Software Modules

Module	Description
PosterHTTPServlet and ClientService	Receives the STB GetPoster request, and returns the requested poster data or exception accordingly.
CSIservice	Receives CSI messages from the VBO, and constructs response messages that it sends to the VBO.
ImageStore	Provides a wrapper of the file system; implements image file store and access, as well as space calculation features.
ImageProcessor	Provides image processing features, including resizing.
TaskManager	Manages image ingestion tasks.
Connection	Communicates with the poster data server in the VBO to download data files. This network-level module is responsible for setting up socket connections. downloading data, and detecting network problems.

Note Cisco CSI supports HTTP and FTP. If other protocols need to be supported in the future, a new type connection must be created and inherited from the Connection class, to implement the download method.

Role of the TaskManager Module

The TaskManager module has a task queue to ensure that ingestion commands do not overwhelm the system.

Note The number of parallel ingestion tasks is configurable. See the "Configuring the Number of Ingestion Tasks" section on page 3-10.

The TaskManager uses the following algorithm:

1. When the system starts up, the task queue is empty.
2. When a job needs to be processed, the TaskManager handles it in one of the following three ways:
   a. Use an available task. If an available task is in the task queue, the TaskManager checks it out and uses it.
   b. Create a task. If the number of tasks is less than the configured number, the TaskManager creates a task, puts it into the task queue, and executes it.
   c. Place job in job queue. The TaskManager puts the job into a job queue until an available task can be used.
3. When the task finishes downloading a poster, it actively checks in with the TaskManager for additional assignments.

The TaskManager uses two basic concepts: *task* and *connection*. Tasks are instantiated, but they do not immediately connect to the target server. When a task is implemented, it creates a connection to the target server and downloads the poster. After the download is complete, the TaskManager tears down the connection to free resources on the target server.

Poster States

A poster may be in one of four states: Pending, Transferring, Completed, or Failed. The transition among those states is as follows:

1. When the TaskManager receives an AddAsset or UpdateAsset command from the VBO, the poster object is created with a Pending state.

Note An asset is the poster art and its associated metadata.

2. When the TaskManager assigns a task to ingest the poster, it changes the state to Transferring.
3. If ingestion is successful, the state becomes Completed.
4. When a poster is transferred, a failure can occur. For example, the VBO storage server may be unreachable, or there could be a network or image resizing failure. In this case, the state is changed to Failed.

The states are detailed below.

- The following applies at the Pending state:
  - If Poster Art Server receives an AddAsset or UpdateAsset request whose version is different from an existing one, it responds with an ExistingOperationInProgress exception.
  - If Poster Art Server receives an AddAsset or UpdateAsset request whose version is the same as an existing one, it responds with a VersionAlreadyExists exception.
  - If Poster Art Server receives a DeleteAsset request, it deletes the asset.
- The following applies at the Transferring state:
  - The VBO should not send an AddAsset or UpdateAsset request. If the VBO becomes aware that the state of a poster has been Transferring for an excessive time, it must first delete the asset, and then add it.

Note The time limitation is determined by the client. Before the file is downloaded from the server, Poster Art Server sets the Transferring state with the protocol to be used (for example, FTP). Depending on the size of the file and the speed of the network, the time of this state is eventually resolved by the success or failure of the download, and the state of the process is changed to either Completed or Failed, as appropriate.

  - If Poster Art Server receives an AddAsset/UpdateAsset request for the same asset that is in Transferring state, it responds with an ExistingOperationInProgress exception.
  - If Poster Art Server receives a DeleteAsset request for the same asset that is in Transferring state, it responds with an ExistingOperationInProgress exception.
- The following applies at the Failed or Completed state:
  - If Poster Art Server receives a SetAssetAction request whose version is the same as that of an existing asset, it responds with a VersionAlreadyExists exception.
  - If Poster Art Server receives a SetAssetAction request whose version is different from that of an existing asset, it deletes the old asset and downloads a new one.
  - If Poster Art Server receives a DeleteAsset request, it deletes the asset.

Implementing Poster Art Server

The following topics are presented:

Installing Poster Art Server

Note CDS Poster Art Server is installed as part of the process that installs CDS Video Navigator. See the "Configuring Video Navigator" section on page 2-5. For complete information on the Cisco CDE110, see the *Cisco Content Delivery Engine 110 Hardware Installation Guide*.

Configuring Poster Art Server to Work with Video Navigator

Note the following implementation issues:

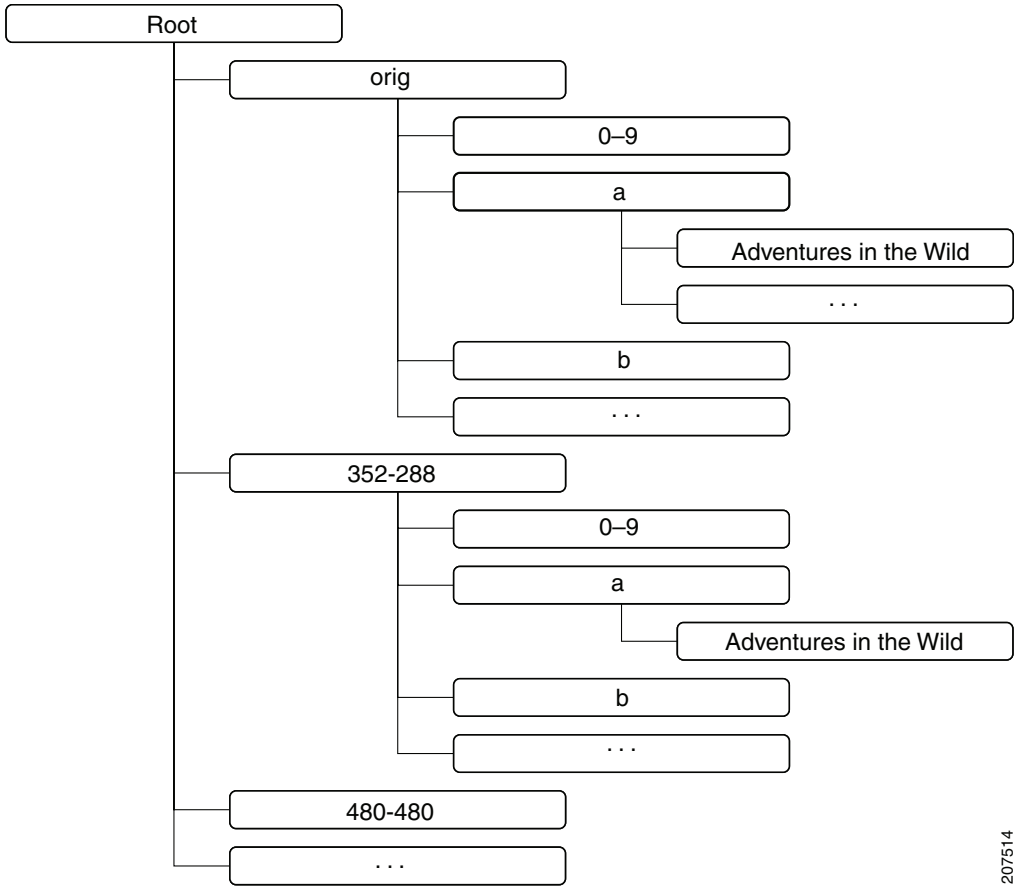
- In the current release, Poster Art Server is designed to coexist with Video Navigator on the same server.
- When requests to Poster Art Server are sent to the IP address of Video Navigator, the web server dispatches separate CSI requests to Video Navigator and to Poster Art Server.
- The interface facing the Video Navigator client can be configured to listen to different ports, so that different STB requests can be handled by different applications.
- Video Navigator stores all data in a database, avsdb, which supports both Video Navigator and Poster Art Server. However, a separate disk partition can be created to store poster data.

Note See the "Storage Database" section on page 3-7.

Understanding the Repository

To allow Poster Art Server to access stored poster files as rapidly as possible, a hierarchical design is used for the poster repository, also known as the image store. The repository is already established and cannot be altered by the operator. The repository hierarchy is shown in Figure 3-1.

Figure 3-1 Repository Hierarchy



- Poster files in the first level, /orig, are separated from resized files. All original ingested files are stored in /orig, no matter what their size is. Resized files are stored in different folders according to their resized dimension.
- The second-level directories are named from a to z and from 0 to 9.
- The poster files are stored in the third level according to the first character of the poster's unique asset ID, which is the name of the file. This prevents duplicate file names if Poster Art Server is used with multiple VBOs in a single deployment.

Note In the case of a failover, the VBO is responsible for synchronizing poster data among multiple servers (see the "VBO-Based Resiliency" section on page 3-8).

Storage Database

Poster information, such as poster ID, poster name, poster version, the number of resized posters, and the name of the resized poster file, are stored in the database. Poster data files and corresponding resized files are stored in the local file system. Table 3-2 details the parameters of the Poster Art Server storage database.

Table 3-2 Poster Art Server Database Parameters

Name	Description	Type	Value	Example
assetId	Unique ID for the poster	String(64)	Any ASCII characters	a188b1dc-376
assetName	Name of the poster (usually the original file name in VBO storage)	String(64)	Any ASCII characters	Mummy3.jpg
Version	Version of the poster	String(16)	Any ASCII characters	75.4.49
Url	URL from which the poster is downloaded	String(1024)	Any ASCII characters	ftp://ingest:ingestqa@192.168.2.22/poster/1.jpg
size	Size of the poster, in bytes	Integer(8)	Any integer	77532
lastModifiedTime	Last time, in seconds, the poster was modified, in decimal time format	Integer(8)	Any integer	1229719854
Width	Width of the original poster, in pixels	Integer(4)	Any integer	352
Height	Height of the original poster, in pixels	Integer(4)	Any integer	288
State	State of the poster, as defined under Value	Integer(2)	Pending (0): asset transfer process waiting to start	3
			Transferring (1): asset being transferred	
			Completed (3): asset transfer completed	
			Failed (4): asset transfer failed	
Reason	Reason for current state	String(64)	Any ASCII characters	Unknown

Designing for Resiliency

Poster Art Server communicates with external components through two interfaces: a CSI interface to the VBO and one to the STB client. (See Figure 1-3 on page 1-5.) For the latter, the optional ACE-based application resiliency feature can be used to implement resiliency. For the former, VBO-based resiliency is required.

ACE-Based Application Resiliency

The basics of configuration for support through Cisco ACE are discussed in the "Configuring for Resiliency with Cisco ACE (Optional)" section on page 2-9.

VBO-Based Resiliency

When Poster Art Server is restarted or a new Poster Art Server server is added into the VoD system, a sign-on is executed (see the "Updating CDS Applications with the Latest Assets upon Sign-On and Registration" section on page 4-23), and the VBO gets the information regarding all posters, compares their versions, and decides whether to download missing poster files. This maintains data synchronization across all servers.

If there is a VBO failure for an unknown reason while a poster is being downloaded, the poster ingestion fails and the download is not resumed—even if the VBO starts up again. The VBO must re-ingest the poster, in which case the state of the poster asset is not complete. Therefore, when the VBO gets the list of all posters, it notices the incomplete download and sends an AddAsset request to Poster Art Server.

Configuring Poster Art Server

To configure Poster Art Server, the operator controls the behavior of the application by editing the web.xml file. See "Preparing to Configure Poster Art Server" section on page 2-6.

Note This file can reside anywhere. Access Video Navigator through the appropriate IP address and upload the file to Video Navigator. For an example of the web.xml file, see the "web.xml" section on page 5-18.

Table 3-3 lists and describes the available parameters for configuring how Poster Art Server operates.

Table 3-3 Configuration Parameters for Poster Art Server Operation

Parameter	Description	Values	Default
PAServerIP	IP address of Poster Art Server CSI application	String	127.0.0.1
PAServerPort	Port of Poster Art Server CSI application	8000–64000	8088
MaxRetry	Maximum number of retries if poster retrieval fails	1–10	3
RetryInterval	Interval between consecutive retries, in seconds	1–60	3
TimeOut	Timeout for VBO connection, in seconds	1–60	3
ResizeOption	Image resize options, in width *x* height	352x288, 480x480. 600x600	See values for ResizeOption in the "web.xml" section on page 5-18. You can add and edit those values as appropriate, keeping the integer format as shown.
ImageStoreRootPath	Root path of the image store (see the "Understanding the Repository" section on page 3-6)	String	/home/isa/paserver/store

In addition to the operation configurations, there is also a system configuration option for tuning Poster Art Server, as shown in Table 3-4 on page 3-9.

Table 3-4 System Configuration Option for Tuning Poster Art Server

Name	Description	Values	Default
TaskNumber	Number of parallel ingestion tasks	1–128	30

Note Before the operator and system configurations can take effect, the system must be restarted. See the "Starting Video Navigator and Verifying System Status" section on page 2-7.

Storing and Resizing Images

Poster Art Server supports the scaling and resizing of images in JPEG format (.jpg) only. (To improve performance, resized images are cached.) The result is an image with 8-bit RGB color components, corresponding to a Windows- or Solaris-style color model, with the colors red, green, and blue packed into integer pixels.

Image resizing works as follows:

- The operator uses the parameter ResizeOption (see Table 3-3 on page 3-8), and the application resizes the poster when the poster is downloaded for the first time. Poster Art Server stores the resized file in an appropriate directory (see "Understanding the Repository" section on page 3-6).
- Alternatively, in the case where ResizeOption is not used, the application does not store a resized file in the image repository.

Note Typically, Poster Art Server does not expect the STB to request a file whose scale has not been configured. To ensure high availability, the operator must use ResizeOption to meet the requirements of different customer equipment—SD TV, HD TV, PC—before poster images are ingested into Poster Art Server.

- When a poster is deleted, the application looks in the "resize" directory and deletes any resized posters.
- When the operator changes a resizing parameter, the change takes effect for new posters only; existing posters are not resized again. Alternatively, the operator updates an existing poster by creating a new version, in which case the application resizes it.

Note The number of posters that can be stored depends on the capacity of the server.

Table 3-5 presents some examples for the number of JPEG poster titles of given sizes that can be stored. Additional storage is needed for each cached image.

Table 3-5 Example Poster Storage Sizes

	Size, KB		
Example	**Low**	**High**	**Average**
320 x 240 JPEG	3.1	26	14.6
80 x 60 JPEG	1.2	7.7	4.5
	Size, MB		
Number of titles	**Low**	**High**	**Average**
1000	5	33	19
5000	22	165	93
10000	43	330	186
25000	106	823	464

Configuring the Number of Ingestion Tasks

You can configure the number of ingestion tasks by changing the value for <*param-value*> in the TaskNum component of the file web.xml. The default is currently 30. The following example changes it to 35.

```
<context-param>
        <param-name>TaskNum</param-name>
        <param-value>35</param-value>
    </context-param>
```