CHAPTER 4

Using the Cisco Content Storage Interface

This chapter discusses the Cisco Content Storage Interface (CSI), and presents the following major topics:

Cisco CSI Overview

The basic functionality of Cisco CSI is introduced in the "Cisco Content Storage Interface Overview" section on page 1-5. The CSI uses detailed metadata for the VOD catalog in conformance with the CableLabs Video-On-Demand Content Specification, Version 1.1.

Cisco CSI Data Types

This section describes the following data types used with the Cisco CSI, presents their schemas, and defines their elements and attributes:

AssetId

AssetId represents the ID of an asset and is unique for each AssetType.

Schema

```
<xs:simpleType name="AssetId">
     <xs:restriction base="xs:string">
<xs:minLength value="1" />
<xs:maxLength value="128" />
<xs:pattern value="[a-zA-Z0-9-_]+" />
     </xs:restriction>
</xs:simpleType>
```

Element	Type	Description
AssetId	xs:string	String from 1 to 128 characters

AssetInfoType

AssetInfoType provides a detailed description of an asset.

Schema

```
<xs:complexType name="AssetInfoType">
    <xs:attribute name="id"         type="tns:AssetId"use="required" />
    <xs:attribute name="state"     type="xs:int"use="required" />
      <xs:attribute name="reason" use="optional">
         <xs:simpleType>
           <xs:restriction base="xs:string">
              <xs:minLength value="1" />
              <xs:maxLength value="64" />
           </xs:restriction>
         </xs:simpleType>
      </xs:attribute>
    <xs:attribute name="version" type="xs:string"use="optional" />
    <xs:attribute name="size"      type="xs:long"use="optional" />
</xs:complexType>
```

Attribute	Type	Description
id	AssetId	Unique ID or name of the asset (for example, funny.jpg) within a specific AssetType
state	xs:int	Possible states: • Pending (0)—Asset transfer process has not started. • Transferring (1)—Asset is being transferred. • Transferring-Playable (2)—Asset is being transferred but a PLAY request can start. Applicable only to AssetType:content. • Completed (3)—Asset transfer has completed. • Failed (4)—Asset transfer has failed. • NonExist (5)—Asset does not exist in the system.
reason	xs:string	Reason why asset is in current state. Not used if state = NonExist. String from 1 to 64 characters.
version	xs:string	Not used if state = NonExist.
size	xs:long	Size of asset in Kbytes. Not used if state = NonExist.

AssetStatusType

AssetStatusType represents the state of an asset.

Schema

```
<xs:complexType name="AssetStatusType">
    <xs:attribute name="id"        type="tns:AssetId" use="required" />
    <xs:attribute name="state"     type="xs:int"      use="required" />
      <xs:attribute name="reason" use="optional">
         <xs:simpleType>
           <xs:restriction base="xs:string">
              <xs:minLength value="1" />
              <xs:maxLength value="64" />
           </xs:restriction>
         </xs:simpleType>
      </xs:attribute>
</xs:complexType>
```

Attribute	Type	Description
id	AssetId	Unique ID or name of the item within a specific AssetType
state	xs:int	Possible states: • Pending (0)—Asset transfer process has not been started. • Transferring (1)—Asset is being transferred. • Transferring-Playable (2)—Asset is being transferred but a PLAY request can start. Applicable only to AssetType:content. • Completed (3)—Asset transfer has completed. • Failed (4)—Asset transfer has failed. • NonExist (5)—Asset does not exist in the system.
reason	xs:string	Reason why the assset is in the current state. Not used if state = NonExist. String from 1 to 64 characters.

AssetType

AssetType represents the type of asset.

Schema

```
<xs:simpleType name="AssetType">
     <xs:restriction base="xs:string">
<xs:enumeration value="catalog" />
<xs:enumeration value="poster" />
<xs:enumeration value="content" />
     </xs:restriction>
</xs:simpleType>
```

Element	Type	Description
AssetType	xs:string	Type of the asset. Possible values: • catalog • poster • content

CallbackAddressType

CallbackAddressType contains information about where the CDS should send the callback message.

Schema

```
<xs:complexType name="CallbackAddressType">
    <xs:attribute name="ipAddress"   type="xs:string"           use="required" />
    <xs:attribute name="port"        type="xs:unsignedShort "   use="required" />
    <xs:attribute name="path"        type="xs:string"           use="required" />
</xs:complexType>
```

Attribute	Type	Description
ipAddress	xs:string	IP address specifying the video backoffice (VBO) to which the CDS should send the callback message. Address can be in dotted decimal notation (IPv4) or colon-separated format (IPv6).
port	xs:unsignedShort	Number of the VBO port to which the CDS should send the callback message
path	xs:string	Full pathname specifying where to send the callback message

CallbackMessageType

CallbackMessageType describes the callback message made to the specified VBO location (CallbackAddressType) after the requested operation is finished. Each callback message contains status for only one asset.

Schema

```
<xs:complexType name="CallbackMessageType">
    <xs:attribute name="type"    type="tns:AssetType"   use="required" />
    <xs:attribute name="id"      type="tns:AssetId"     use="required" />
    <xs:attribute name="status"  type="xs:string"       use="required" />
    <xs:attribute name="reason"  type="xs:string"       use="required" />
</xs:complexType>
```

Attribute	Type	Description
type	AssetId	Type of asset
id	AssetType	Unique ID or name of the item within a specific AssetType
status	xs:string	Result of the operation. Possible values: • succeed • failed
reason	xs:string	Reason for the result

ErrorResponseType

ErrorResponseType describes the error message that is returned if there is problem servicing a request.

Schema

```
<xs:complexType name="ErrorResponseType">
       <xs:attribute name="code" type="xs:string" />
       <xs:attribute name="reason" type="xs:string" />
</xs:complexType>
```

Attribute	Type	Description
code	xs:string	Error code (see Table 4-1)
reason	xs:string	Reason for the error (see Table 4-1)

Table 4-1 lists error codes and associated reasons.

Table 4-1 CSI Error Codes and Reasons

Error Code	Reason
InvalidAssetType	Asset type specified is invalid
InvalidId	ID is in invalid format
VersionAlreadyExists	Version of this asset already exists in the system
VersionDoesNotExist	Version specified does not exist in the system
InvalidTransferProtocol	Invalid transfer protocol specified
InvalidAddressType	Invalid callback address specified
InternalError	Internal processing error
VBORepositoryError	VBO repository error, cannot access data
ExistingOperationInProgress	Operation on the asset is currently in progress
InvalidXMLFormat	XML data in the message is incorrectly formed
InvalidContentFormat	Content (poster and movie) format is not valid

TransferProtocolType

TransferProtocolType describes the protocol to be used for fetching assets.

Schema

```
<xs:complexType name="TransferProtocolType">
    <xs:attribute name="protocol"     type="xs:string"          use="required" />
    <xs:attribute name="ipAddress"    type="xs:string"          use="required" />
    <xs:attribute name="port"         type="xs:unsignedShort"   use="required" />
    <xs:attribute name="userName"     type="xs:string"          use="optional" />
    <xs:attribute name="password"     type="xs:string"          use="optional" />
    <xs:attribute name="path"         type="xs:string"          use="required" />
 </xs:complexType>
```

Attribute	Type	Description
protocol	xs:string	Possible values: • FTP • HTTP GET
ipAddress	xs:string	IP address in dotted decimal notation (IPv4) or colon-separated format (IPv6)
port	xs:unsignedShort	Port number
username	xs:string	Used for certain protocols (FTP)
password	xs:string	Used for certain protocols (FTP)
path	xs:string	Full pathname to the asset, for example, ~/posters/example_1.jpg

Cisco CSI Requests and Responses

This section describes the following requests and responses for Cisco CSI:

Note The namespace for the above requests and responses is com.cisco.csi.

DeleteAsset

DeleteAsset requests the deletion of an asset.

Note The request is carried out asynchronously. A requester wanting to be notified when the operation is completed should register for a callback notification by using the callback element.

Message Summary

Message	XML Element
Request	DeleteAssetRequest
Response	DeleteAssetResponse

Request

Element: DeleteAssetRequest

Schema:

```
<xs:element name="DeleteAssetRequest">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="type" type="tns:AssetType" />
        <xs:element name="id" type="tns:AssetId" />
        <xs:element name="version" type="xs:string" />
        <xs:element minOccurs="0" name="callback" type="tns:CallbackAddressType" />
      </xs:sequence>
    </xs:complexType>
</xs:element>
```

Element	Type	Description
type	AssetType	Type of asset
id	AssetId	Unique ID of the asset. Maximum length is set to 128.
version	xs:string	Version of the asset
callback	CallbackAddressType	Callback address. Notification is made to the address provided after the delete asset process is finished. This parameter is optional. Without it, no notification is made. **Note** The requester can request only one callback location.

Example Request:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<DeleteAssetRequest xmlns="com.cisco.csi">
    <type>poster</type>
    <id>1234567</id>
    <version>1.1</version>
</DeleteAssetRequest>
```

Response

Element: DeleteAssetResponse

Schema:

```
<xs:element name="DeleteAssetResponse">
     <xs:complexType>
      <xs:sequence>
        <xs:element name="status" type="tns:AssetStatusType" />
      </xs:sequence>
     </xs:complexType>
</xs:element>
```

Element	Type	Description
status	AssetStatusType	Status of an asset. Possible values: • will get state—pending • exists—false with this operation

Example Response:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<DeleteAssetResponse xmlns="com.cisco.csi">
    <status reason="Pending" state="0" exists="false" id="1234567"/>
</DeleteAssetResponse>
```

Error Codes

- InvalidAssetType
- InvalidXMLFormat
- InvalidContentFormat
- InvalidId
- VersionDoesNotExist
- InvalidTransferProtocol
- InvalidCallbackAddress
- InternalError

Note For definitions of the above error codes, see Table 4-1 on page 4-7.

GetAssetCount

GetAssetCount returns the number of the requested asset type that exists in the system.

Message Summary

Message	XML Element
Request	GetAssetCountRequest
Response	GetAssetCountResponse

Request

Element: GetAssetCountRequest

Schema:

```
<xs:element name="GetAssetCountRequest">
     <xs:complexType>
          <xs:sequence>
              <xs:element name="type" type="tns:AssetType" />
          </xs:sequence>
     </xs:complexType>
</xs:element>
```

Element	Type	Description
type	AssetType	Type of asset

Example Request:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<GetAssetCountRequest xmlns="com.cisco.csi">
    <type>poster</type>
</GetAssetCountRequest>
```

Response

Element: GetAssetCountResponse

Schema:

```
<xs:element name="GetAssetCountResponse">
      <xs:complexType>
      <xs:sequence>
        <xs:element name="count" type="xs:int" />
      </xs:sequence>
      </xs:complexType>
</xs:element>
```

Element	Type	Description
count	xs:int	Number of assets

Example Response:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<GetAssetCountResponse xmlns="com.cisco.csi">
    <count>2</count>
</GetAssetCountResponse>
```

Error Codes

- InvalidAssetType
- Invalid XMLFormat
- InternalError

Note For definitions of the above error codes, see Table 4-1 on page 4-7.

GetAssetInventoryList

GetAssetInventoryList returns an inventory list for the requested asset type.

Message Summary

Message	XML Element
Request	GetAssetInventoryListRequest
Response	GetAssetInventoryListResponse

Request

Element: GetAssetInventoryListRequest

Schema:

```
<xs:element name="GetAssetInventoryListRequest">
      <xs:complexType>
      <xs:sequence>
        <xs:element name="type" type="tns:AssetType" />
      </xs:sequence>
      </xs:complexType>
</xs:element>
```

Element	Type	Description
type	AssetType	Type of asset

Example Request:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<GetAssetInventoryListRequest xmlns="com.cisco.csi">
    <type>poster</type>
</GetAssetInventoryListRequest>
```

Response

Element: GetAssetInventoryListResponse

Schema:

```
<xs:element name="GetAssetInventoryListResponse">
    <xs:complexType>
      <xs:sequence>
            <xs:element minOccurs="0" maxOccurs="unbounded" name="info"
type="tns:AssetInfoType" />
      </xs:sequence>
    </xs:complexType>
</xs:element>
```

Element	Type	Description
type	AssetInfoType	Information about the type of asset

Example Response:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<GetAssetInventoryListResponse xmlns="com.cisco.csi">
    <info size="84000" version="1.1" reason="Pending" state="0" id="1234567"/>
    <info size="44000" version="1.1" reason="Completed" state="3" id="7654321"/>
</GetAssetInventoryListResponse>
```

Error Codes

- InvalidAssetType
- InvalidXMLFormat
- InternalError

Note For definitions of the above error codes, see Table 4-1 on page 4-7.

GetAssetsInfo

GetAssetsInfo returns the description of the requested asset.

Message Summary

Message	XML Element
Request	GetAssetsInfoRequest
Response	GetAssetsInfoResponse

Request

Element: GetAssetsInfoRequest

Schema:

```
<xs:element name="GetAssetsInfoRequest">
      <xs:complexType>
      <xs:sequence>
        <xs:element name="type" type="tns:AssetType" />
        <xs:element maxOccurs="unbounded" name="id" type="tns:AssetId" />
      </xs:sequence>
      </xs:complexType>
</xs:element>
```

Element	Type	Description
type	AssetType	Type of asset
id	AssetId	Unique ID of the asset. Maximum length is set to 128.

Example Request:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<GetAssetsInfoRequest xmlns="com.cisco.csi">
    <type>poster</type>
    <id>1234567</id>
    <id>7654321</id>
</GetAssetsInfoRequest>
```

Response

Element: GetAssetsInfoResponse

Schema:

```
<xs:element name="GetAssetsInfoResponse">
      <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" name="info" type="tns:AssetInfoType" />
      </xs:sequence>
      </xs:complexType>
</xs:element>
```

Element	Type	Description
info	AssetInfoType	Information about the asset

Example Response:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<GetAssetsInfoResponse xmlns="com.cisco.csi">
    <info size="84000" version="1.1" reason="Pending" state="0" id="1234567"/>
    <info size="44000" version="1.1" reason="Completed" state="3" id="7654321"/>
</GetAssetsInfoResponse>
```

Error Codes

- InvalidAssetType
- InvalidXMLFormat
- InvalidId
- InternalError

Note For definitions of the above error codes, see Table 4-1 on page 4-7.

GetAssetsStatus

GetAssetsStatus returns the state of the requested asset.

Message Summary

Message	XML Element
Request	GetAssetsStatusRequest
Response	GetAssetsStatusResponse

Request

Element: GetAssetsStatusRequest

Schema:

```
<xs:element name="GetAssetsStatusRequest">
      <xs:complexType>
      <xs:sequence>
        <xs:element name="type" type="tns:AssetType" />
        <xs:element maxOccurs="unbounded" name="id" type="tns:AssetId" />
      </xs:sequence>
      </xs:complexType>
</xs:element>
```

Element	Type	Description
type	AssetType	The type of asset
id	AssetId	Unique ID of the asset. Maximum length is set to 128.

Example Request:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<GetAssetsStatusRequest xmlns="com.cisco.csi">
    <type>poster</type>
    <id>1234567</id>
    <id>7654321</id>
</GetAssetsStatusRequest>
```

Response

Element: GetAssetsStatusResponse

Schema:

```
<xs:element name="GetAssetsStatusResponse">
     <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" name="status" type="tns:AssetStatusType" />
      </xs:sequence>
     </xs:complexType>
</xs:element>
```

Element	Type	Description
status	AssetStatusType	Status of an asset. Possible values: • If exists = false, the asset is NOT in the CDS system. • If exists = true, the asset is in the CDS system.

Example Response:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<GetAssetsStatusResponse xmlns="com.cisco.csi">
    <status reason="Pending" state="0" exists="true" id="1234567"/>
    <status reason="Completed" state="3" exists="false" id="7654321"/>
</GetAssetsStatusResponse>
```

Error Codes

- InvalidAssetType
- InvalidXMLFormat
- InvalidId
- InternalError

Note For definitions of the above error codes, see Table 4-1 on page 4-7.

GetAvailableSpace

GetAvailableSpace returns the quantity of space available for an asset type.

Message Summary

Message	XML Element
Request	GetAvailableSpaceRequest
Response	GetAvailableSpaceResponse

Request

Element: GetSpaceAvailableRequest

Schema:

```
<xs:element name="GetAvailableSpaceRequest">
      <xs:complexType>
      <xs:sequence>
        <xs:element name="type" type="tns:AssetType" />
      </xs:sequence>
      </xs:complexType>
</xs:element>
```

Element	Type	Description
type	AssetType	Type of asset

Example Request:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<GetAvailableSpaceRequest xmlns="com.cisco.csi">
    <type>poster</type>
</GetAvailableSpaceRequest>
```

Response

Element: GetAvailableSpaceResponse

Schema:

```
<xs:element name="GetAvailableSpaceResponse">
    <xs:complexType>
         <xs:sequence>
             <xs:element name="totalSpace" type="xs:long" />
           <xs:element name="freeSpace" type="xs:long" />
         </xs:sequence>
    </xs:complexType>
</xs:element>
```

Element	Type	Description
totalSpace	xs:long	Total space in kilobytes
freeSpace	xs:long	Free space in kilobytes

Example Response:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<GetAvailableSpaceResponse xmlns="com.cisco.csi">
    <totalSpace>200000</totalSpace>
    <freeSpace>100000</freeSpace>
</GetAvailableSpaceResponse>
```

Error Codes

- InvalidAssetType
- InvalidXMLFormat
- InternalError

Note For definitions of the above error codes, see Table 4-1 on page 4-7.

SetAssetAction

SetAssetAction requests the ingestion or update of an asset. The CDS component uses the protocol provided to fetch the asset.

Note The request is carried out asynchronously. A requester wanting to be notified when the operation is completed should register for a callback notification by using the callback element.

Message Summary

Message	XML Element
Request	SetAssetRequest
Response	SetAssetResponse

Request

Element: SetAssetRequest

Schema:

```
<xs:element name="SetAssetRequest">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="type"    type="tns:AssetType" />
        <xs:element name="id"        type="tns:AssetId" />
        <xs:element name="version"  type="xs:string" />
        <xs:element name="protocol" type="tns:TransferProtocolType" />
        <xs:element minOccurs="0" name="callback"  type="tns:CallbackAddressType" />
      </xs:sequence>
    </xs:complexType>
</xs:element>
```

Element	Type	Description
type	AssetType	Type of asset
id	AssetId	Unique ID of the asset. Maximum length is set to 128.
version	xs:string	Version of the asset
protocol	TransferProtocolType	Transfer protocol for getting the image
callback	CallbackAddressType	Callback address. Notification is made to the address provided after the add asset process is finished. This parameter is optional. Without it, no notification is made. **Note** The requester can request only one callback location.

Example Request:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<SetAssetRequest xmlns="com.cisco.csi">
    <type>poster</type>
    <id>1234567</id>
    <version>1.1</version>
    <protocol path="/poster" password="ingestqa" userName="ingest" port="21"
ipAddress="192.168.1.88" protocol="ftp"/>
    <callback path="/callback" port="1234" ipAddress="192.168.1.100"/>
</SetAssetRequest>
```

Response

Element: SetAssetResponse

Schema:

```
<xs:element name="SetAssetResponse">
     <xs:complexType>
      <xs:sequence>
        <xs:element name="status"  type="tns:AssetStatusType" />
      </xs:sequence>
     </xs:complexType>
</xs:element>
```

Element	Type	Description
status	AssetStatusType	Status of an asset. Possible values: • will get state—pending • exists—true with this operation

Example Response:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<SetAssetResponse xmlns="com.cisco.csi">
    <status reason="Pending" state="0" exists="false" id="1234567"/>
</SetAssetResponse>
```

Error Codes

- InvalidAssetType
- InvalidXMLFormat
- InvalidContentFormat
- InvalidId
- VersionAlreadyExist
- InvalidTransferProtocol
- InvalidCallbackAddress
- ExistingOperationInProgress
- VBORepositoryError
- InternalError

Note For definitions of the above error codes, see Table 4-1 on page 4-7.

Using the Cisco CSI to Update CDS Applications

This section provides the following detailed use cases for the Cisco Content Storage Interface:

Updating CDS Applications with the Latest Assets upon Sign-On and Registration

This section describes a typical interaction used for updating Cisco CDS applications (Video Navigator or Poster Art Server) when the application first signs on or registers with the VBO. Requests from the VBO for the CDS application to add or delete an asset are performed asynchronously. If the VBO requires notification that the add or delete operation has completed, the VBO must provide callback information in the add or delete request.

The following is a sequence of typical CSI API interactions between the VBO and the CDS application when the application registers (signs on) with the VBO.

1. When the application starts up, it sends a **SignOn** request to the VBO.
2. The VBO uses **GetAssetInventoryList** to get an inventory list of the appropriate asset type (for example, catalog or poster) from the application. The response from the application includes asset information such as ID, current state, version, and size.
3. In a loop operation, the VBO uses **DeleteAsset** to request that the application delete assets that are not part of the internal list of the VBO. Using the asset ID specified in the request, the application starts processing the delete request and then sends a **DeleteAssetResponse** to the VBO.
4. Optionally, as part of the loop operation, the application performs a callback to notify the VBO that the delete operation is completed. If a callback is used, the details of how to perform the callback are provided by the VBO in a **DeleteAssetRequest**.
5. In a loop operation, the VBO uses **SetAsset** to request that the application add missing assets. The application starts processing the request and then sends a **SetAssetResponse** to the VBO.
6. The application fetches the asset by using information provided in the VBO **SetAssetRequest**, such as the protocol, IP address, and port number.
7. Optionally, as part of the loop operation, the application performs a callback to notify the VBO that the add operation is completed. If a callback is used, the details of how to perform the callback are provided by the VBO in the **SetAssetRequest**.
8. In a loop operation, the VBO uses **SetAsset** to request that the application update assets that are out of date. The application starts processing the update request and then sends a **SetAssetResponse** to the VBO. (The update process is very similar to the add process, but includes deleting each out-of-date asset.)
9. The application fetches the updated asset by using information provided in the **SetAssetRequest**.
10. Optionally, as part of the loop operation, the component performs a callback to notify the VBO that the update operation is completed. If a callback is used, the details of how to perform the callback are provided by the VBO in the **SetAssetRequest**.

Updating CDS Applications with the Catalog and Poster Art

This section describes a typical interaction used in updating the catalog in Video Navigator and updating the poster art in Poster Art Server. The following conditions must be met before an update can take place:

- Filename of the asset detail data is *AssetID*.xml, where *AssetID* is a unique asset ID.
- Filename of the product detail data is *ProductID*.xml, where *ProductID* is a unique product ID.
- Filename of the poster is defined in *AssetID*.xml.
- When the asset detail information is updated in the asset detail file, the asset version is updated in the catalog file.
- When the product detail information of the asset is updated in the product detail file, the asset version is updated in the catalog file.
- Asset detail files, product detail files, and posters specified by one catalog file are located in the same directory as that catalog file.
- Transport protocol of the asset detail files, the product detail files, and the posters is the same as the transport protocol for the catalog.
- In a poster update, because the poster's version is embedded in the response of the GetAssetInventoryList request, the VBO can determine whether posters need to be updated. It is assumed that Poster Art Server always downloads the poster when it receives an AddAsset request.

The following is a sequence of typical interactions between the VBO and the CDS application during the catalog and poster art updates:

1. The VBO uses SetAssetAction to request that Video Navigator add the catalog. As part of this request, the VBO provides the asset type, ID, and version, as well as the protocol to use for the transfer. If a callback is used, the details for the callback are provided by the VBO in the SetAssetAction request.
2. Video Navigator downloads the catalog through the transport protocol defined in the SetAssetAction request.
3. Video Navigator parses the catalog file, and then compares the asset version with the version that currently exists in the Video Navigator system, as follows:
   a. If the asset version in the catalog is newer than the one in the Video Navigator system, Video Navigator downloads the asset. If the asset version is not newer than the one already in the system, Video Navigator processes the next asset.
   b. If the asset version in the catalog is newer than the one in the system, Video Navigator downloads *AssetID*.xml first, parses the metadata, and then stores the metadata in the database for future retrieval.
   c. If the asset version in the catalog is newer than the one in the system, Video Navigator also downloads, parses, and stores *ProductID*.xml.
4. After processing all of the assets, Video Navigator uses a callback to notify the VBO that ingestion is completed. The use of a callback is optional.

The sequence for updating a poster art asset on Poster Art Server is as follows:

Note To ensure that the most recent poster art is available, the VBO checks the poster art version before using SetAssetAction.

1. The VBO uses SetAssetAction to request that Poster Art Server add a poster. As part of SetAssetAction, the VBO provides the asset type, ID, and version, as well as the protocol to use for the transfer. If a callback is used, the details for the callback are provided by the VBO in the SetAssetAction request.
2. Poster Art Server downloads the poster through the transport protocol defined in the SetAssetAction request.
3. After the poster is downloaded and resized, Poster Art Server uses a callback to notify the VBO that ingestion is completed. The use of a callback is optional.

VBO Registration APIs

This section documents a reference registration interface provided by VBO vendors. Cisco CDS applications such as Video Navigator and Poster Art Server use this interface to register and deregister themselves with the VBO. The following registration APIs are presented:

- SignOn, page 4-25
- SignOff, page 4-27
- XSD for SignOn and SignOff Request and Response Pair, page 4-28

Note The namespace for the VBO registration API is defined by the VBO.

SignOn

Message Summary

Message	XML Element
Request	SignOnRequest
Response	SignOnResponse

Request

Element: SignOnRequest

Schema:

```
<element name="SignOnRequest">
        <attribute name="ipAddress"  type="xs:string"  use="required"/>
        <attribute name="port"  type="xs:string"  use="required"/>
        <attribute name="name"  type="xs:string"  use="optional"/>
        <attribute name="type"  type="AssetType"  use="required"/>
</element>
```

Element	Type	Description
ipAddress	xs:string	Dotted-decimal notation (IPv4) or colon-separated (IPv6) format
port	xs:int	Port number
name	xs:string	Name of the CDS application
type	AssetType	Asset type of interest to the CDS application

Example Request:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<SignOnRequest xmlns="com.cisco.csi" type="poster" name="poster art server" port="6666"
ipAddress="192.168.1.66"/>
```

Response

Element: SignOnResponse

Schema:

```
<element name="SignOnResponse" />
   <complexType>
      <sequence>
            <element name="status" type="xs:string" minOccurs="1" maxOccurs="1">
<simpleType>
                  <restriction base="xs:string" />
                           <enumeration  value="Success" />
                           <enumeration  value="Failed" />
                           <enumeration  value="InvaidParameters" />
                   </restriction />
          </simpleType>
</element>
      </sequence>
   </complexType>
</element>
```

Element	Type	Description
status	xs:string	Status of the response to the SignOn request. Possible responses: • Success • Failed • Invalid parameters

Example Response:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<SignOnResponse xmlns="com.cisco.csi">
    <status>Success</status>
</SignOnResponse>
```

SignOff

Message Summary

Message	XML Element
Request	SignOffRequest
Response	SignOffResponse

Request

Element: SignOffRequest

Schema:

```
<element name="SignOffRequest">
        <attribute name="ipAddress"  type="xs:string"  use="required"/>
        <attribute name="port"  type="xs:int"  use="required"/>
        <attribute name="name"  type="xs:string"  use="optional"/>
</element>
```

Element	Type	Description
ipAddress	xs:string	Dotted-decimal notation (IPv4) or colon-separated (IPv6) format
port	xs:int	Port number
name	xs:string	Name of the CDS application

Example Request:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<SignOffRequest xmlns="com.cisco.csi" name="poster art server" port="6666"
ipAddress="192.168.1.66"/>
```

Response

Element: SignOffResponse

Schema:

```
<element name="SignOffResponse" />
   <complexType>
      <sequence>
            <element name="status" type="xs:string" minOccurs="1" maxOccurs="1">
<simpleType>
                  <restriction base="xs:string" />
                           <enumeration  value="Success" />
                           <enumeration  value="Failed" />
                           <enumeration  value="InvaidParameters" />
                   </restriction />
          </simpleType>
</element>
      </sequence>
   </complexType>
</element>
```

Element	Type	Description
status	xs:string	Status of the response to the SignOff request. Possible responses: • Success • Failed • Invalid parameters

Example Response:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<SignOffResponse xmlns="com.cisco.csi">
    <status>Success</status>
</SignOffResponse>
```

XSD for SignOn and SignOff Request and Response Pair

The following schema is a sample XSD for the SignOn and SignOff Request and Response pair.

Schema

```
<xs:element name="SignOnRequest">
    <xs:complexType>
      <xs:attribute name="ipAddress" type="xs:string" />
      <xs:attribute name="port" type="xs:int" />
      <xs:attribute name="name" type="xs:string" use="optional" />
      <xs:attribute name="type" type="tns:AssetType" />
    </xs:complexType>
  </xs:element>
  <xs:element name="SignOnResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="status" type="xs:string" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="SignOffRequest">
    <xs:complexType>
      <xs:attribute name="ipAddress" type="xs:string" />
      <xs:attribute name="port" type="xs:int" />
      <xs:attribute name="name" type="xs:string" />
    </xs:complexType>
  </xs:element>
  <xs:element name="SignOffResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="status" type="xs:string" />
      </xs:sequence>
    </xs:complexType>
</xs:element>
```

Required ADI Metadata in Catalog

This section provides information related to the required Asset Distribution Interface (ADI) metadata in the catalog. The ADI is the means by which assets, both content and metadata describing the content, are transported from the provider to the asset management system (AMS). The ADI specification defines the logical organization of metadata structures (Group Asset, Metadata Asset, Content Asset), as well as the operations that put content assets in context with one or more service offerings.

Note See the following CableLabs documents:

CableLabs Asset Distribution Interface Specification Version 1.1

CableLabs Video-On-Demand Content 1.0 Specification

Table 4-2 shows the specific ADI metadata in the catalog, with abbreviations and definitions as listed in Table 4-3 on page 4-30.

Table 4-2 Required ADI Metadata

Name	Type	Specification	Required by ADI	Required by VN
Actors	Title	MOD/SVOD	O	Y
Actors_Display	Title	MOD/SVOD	O	Y
Advisories	Title	MOD/SVOD	O	N
Audience	Title	MOD/SVOD	O	Y
Audio_Type	Movie	MOD/SVOD	R	Y
Audio_Type	Preview	MOD/SVOD	R	N
Chapter	Title	MOD/SVOD	O	N
Country_of_Origin	Title	MOD/SVOD	O	N
Director	Title	MOD/SVOD	O	Y
Display_Run_Time	Title	MOD/SVOD	R	N
Dubbed_Langauges	Movie	MOD/SVOD	O	N
Dubbed_Langauges	Preview	MOD/SVOD	O	N
Episode_ID	Title	MOD/SVOD	O	Y
Episode_Name	Title	MOD/SVOD	O	Y
Genre	Title	MOD/SVOD	O	Y
HDContent	Movie	MOD/SVOD	R	Y
HDContent	Preview	MOD/SVOD	O	N
ISAN	Title	MOD/SVOD	O	N
Languages	Movie	MOD/SVOD	O	N
Languages	Preview	MOD/SVOD	O	N
MSORating	Title	MOD/SVOD	O	N
Producers	Title	MOD/SVOD	O	Y

Table 4-2 Required ADI Metadata (continued)

Name	Type	Specification	Required by ADI	Required by VN
Provider	Movie	AMS	R	Y
Rating	Title	MOD/SVOD	R	Y
Rating	Preview	MOD/SVOD	R	N
Run_Time	Title	MOD/SVOD	R	Y
Run_Time	Preview	MOD/SVOD	R	Y
Screen_Format	Movie	MOD/SVOD	O	Y
Screen_Format	Preview	MOD/SVOD	O	N
Season_Finale	Title	MOD/SVOD	O	Y
Season_Premiere	Title	MOD/SVOD	O	N
Studio	Title	MOD/SVOD	O	Y
Subtitle_Languages	Movie	MOD/SVOD	O	N
Subtitle_Languages	Preview	MOD/SVOD	O	N
Summary_Long	Title	MOD/SVOD	O	N
Summary_Medium	Title	MOD/SVOD	O	N
Summary_Short	Title	MOD/SVOD	R	Y
Title	Title	MOD/SVOD	R	Y
Title_Brief	Title	MOD/SVOD	R	Y
Title_Sort_Name	Title	MOD/SVOD	O	N
Writer_Display	Title	MOD/SVOD	O	Y
Year	Title	MOD/SVOD	O	Y
Year	Preview	MOD/SVOD	O	Y

Table 4-3 Abbreviations and Definitions for ADI Metadata

Abbreviation	Definition
MOD	Movie on Demand
SVOD	Subscriber Video on Demand
AMS	Asset Management System
ADI	Asset Distribution Interface
VN	Cisco CDS Video Navigator
O	Optional
R	Required
Y	Yes
N	No

Cisco CSI XSD

This section provides the complete Cisco CSI schema. XML is used for data exchange, and object representation is documented in "XML Schema Definition (XSD)."

Schema

```
<?xml version="1.0" encoding="utf-8" ?>
<xs:schema xmlns:tns="com.cisco.csi" elementFormDefault="qualified"
targetNamespace="com.cisco.csi" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:simpleType name="AssetType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="catalog" />
      <xs:enumeration value="poster" />
      <xs:enumeration value="content" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="AssetId">
    <xs:restriction base="xs:string">
      <xs:maxLength value="128" />
      <xs:minLength value="1" />
      <xs:pattern value="[a-zA-Z0-9-_]+" />
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="AssetInfoType">
    <xs:attribute name="id" type="tns:AssetId" use="required" />
    <xs:attribute name="state" type="xs:int" use="required" />
    <xs:attribute name="reason" use="optional">
       <xs:simpleType>
          <xs:restriction base="xs:string">
             <xs:minLength value="1" />
             <xs:maxLength value="64" />
          </xs:restriction>
       </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="version" type="xs:string" use="optional" />
    <xs:attribute name="size" type="xs:long" use="optional" />
  </xs:complexType>
  <xs:complexType name="AssetStatusType">
    <xs:attribute name="id" type="tns:AssetId" use="required" />
    <xs:attribute name="state" type="xs:int" use="required" />
    <xs:attribute name="reason" use="optional">
       <xs:simpleType>
          <xs:restriction base="xs:string">
             <xs:minLength value="1" />
             <xs:maxLength value="64" />
          </xs:restriction>
       </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
  <xs:complexType name="TransferProtocolType">
    <xs:attribute name="protocol" type="xs:string" use="required" />
    <xs:attribute name="ipAddress" type="xs:string" use="required" />
    <xs:attribute name="port" type="xs:unsignedShort" use="required" />
    <xs:attribute name="userName" type="xs:string" use="optional" />
    <xs:attribute name="password" type="xs:string" use="optional" />
    <xs:attribute name="path" type="xs:string" use="required" />
  </xs:complexType>
  <xs:complexType name="CallbackAddressType">
    <xs:attribute name="ipAddress" type="xs:string" use="required" />
    <xs:attribute name="port" type="xs:unsignedShort" use="required" />
    <xs:attribute name="path" type="xs:string" use="required" />
```

```
  </xs:complexType>
  <xs:complexType name="CallbackMessageType">
    <xs:attribute name="type" type="tns:AssetType" use="required" />
    <xs:attribute name="id" type="tns:AssetId" use="required" />
    <xs:attribute name="status" type="xs:string" use="required" />
    <xs:attribute name="reason" type="xs:string" use="required" />
  </xs:complexType>
  <xs:complexType name="ErrorResponseType">
    <xs:attribute name="code" type="xs:string" />
    <xs:attribute name="reason" type="xs:string" />
  </xs:complexType>
  <xs:element name="GetAssetCountRequest">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="type" type="tns:AssetType" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="GetAssetCountResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="count" type="xs:int" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="GetAssetInventoryListRequest">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="type" type="tns:AssetType" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="GetAssetInventoryListResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" maxOccurs="unbounded" name="info"
type="tns:AssetInfoType" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="GetAvailableSpaceRequest">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="type" type="tns:AssetType" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="GetAvailableSpaceResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="totalSpace" type="xs:long" />
        <xs:element name="freeSpace" type="xs:long" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="SetAssetRequest">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="type" type="tns:AssetType" />
        <xs:element name="id" type="tns:AssetId" />
        <xs:element name="version" type="xs:string" />
        <xs:element name="protocol" type="tns:TransferProtocolType" />
        <xs:element minOccurs="0" name="callback" type="tns:CallbackAddressType" />
      </xs:sequence>
```

```
    </xs:complexType>
  </xs:element>
  <xs:element name="SetAssetResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="status" type="tns:AssetStatusType" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="DeleteAssetRequest">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="type" type="tns:AssetType" />
        <xs:element name="id" type="tns:AssetId" />
        <xs:element name="version" type="xs:string" />
        <xs:element minOccurs="0" name="callback" type="tns:CallbackAddressType" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="DeleteAssetResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="status" type="tns:AssetStatusType" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="GetAssetsStatusRequest">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="type" type="tns:AssetType" />
        <xs:element maxOccurs="unbounded" name="id" type="tns:AssetId" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="GetAssetsStatusResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" name="status" type="tns:AssetStatusType" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="GetAssetsInfoRequest">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="type" type="tns:AssetType" />
        <xs:element maxOccurs="unbounded" name="id" type="tns:AssetId" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="GetAssetsInfoResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" name="info" type="tns:AssetInfoType" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```