CableLabs® Asset Distribution Interface Specification Version 1.1

MD-SP-ADI1.1-I01-020927

Issued

Notice

Document Status Sheet

Document Control Number:	MD-SP-ADI1.1-I01-020927			
Document Title:	CableLabs® Asset Distribution Interface Specification Version 1.1			
Revision History:	I01 – Released 9/27/02			
Date:	September 27, 2002			
Status:	~~WIP~~	~~Draft~~	Issued	~~Released~~
Distribution Restrictions:	~~Author Only~~	~~CL/Member~~	~~CL/ Member/ Vendor~~	Public

Key to Document Status Codes:

Work in Progress	An incomplete document, designed to guide discussion and generate feedback, that may include several alternative requirements for consideration.
Draft	A document in specification format considered largely complete, but lacking review by Members and vendors. Drafts are susceptible to substantial change during the review process.
Issued	A stable document, which has undergone rigorous member and vendor review and is suitable for product design and development, cross-vendor interoperability, and for certification testing.
Closed	A static document, reviewed, tested, validated, and closed to further engineering change requests to the specification through CableLabs.

Table of Contents

List of Figures

List of Tables

1 INTRODUCTION

1.1 Purpose

The Asset Distribution Interface is the means by which assets (content as well as metadata describing that content) are transferred from a provisioning Asset Management System. In addition, it provides mechanisms by which 'block updates' may be made to previously distributed assets. These block updates could include new/data element, content replacement, adding new child assets to a previously received asset and deleting assets.

1.2 Scope

Version 1.0 of this specification was concerned with asset distribution using DLT tape as the medium. This version allows for other distribution media. This version of ADI defines distribution of assets over a network interface. Assets are contained in a package that is moved from an Asset Distribution System (ADS) to an Asset Management System (AMS). The methods by which assets are distributed within the ADS are beyond the scope of this specification.

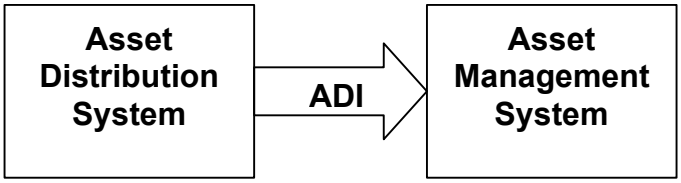

Figure 1 – Asset Distribution

1.3 Requirements

Throughout this document, the words that are used to define the significance of particular requirements are capitalized. These words are:

"MUST"	This word or the adjective "REQUIRED" means that the item is an absolute requirement of this specification.
"MUST NOT"	This phrase means that the item is an absolute prohibition of this specification.
"SHOULD"	This word or the adjective "RECOMMENDED" means that there may exist valid reasons in particular circumstances to ignore this item, but the full implications should be understood and the case carefully weighed before choosing a different course.
"SHOULD NOT"	This phrase means that there may exist valid reasons in particular circumstances when the listed behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
"MAY"	This word or the adjective "OPTIONAL" means that this item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because it enhances the product, for example; another vendor may omit the same item.

2 REFERENCES

2.1 Normative References

In order to claim compliance with this specification, it is necessary to conform to the following standards and other works as indicated, in addition to the other requirements of this specification. Notwithstanding, intellectual property rights may be required to use or implement such normative references.

[1] Extensible Markup Language (XML) Version 1.0, World Wide Web Consortium

[2] IETF RFC 1738, Uniform Resource Locators (URL), December 1994, www.ietf.org

2.2 Informative References

[3] CableLabs Video-On-Demand Content Specification Version 1.1, MD-SP-VOD-CONTENT1.1-I01-020927, September 27, 2002

[4] The Common Object Request Broker: Architecture and Specification, OMG, CORBA services: Common Object Services Specification, OMG

[5] IETF RFC 959, File Transfer Protocol (FTP), October 1985

[6] IETF RFC 1945, Hypertext Transfer Protocol -- HTTP/1.0, May 1996

[7] Pegasus 1.0 RFP, Time Warner Cable

[8] Pegasus 2.0 RFP. Time Warner Cable

[9] Pegasus ADI 2.0, Time Warner Cable

[10] Pegasus Interactive Services Architecture 1.1, Time Warner Cable

[11] Pegasus Three Letter Acronyms Version 1.0, Time Warner Cable

2.3 Reference Acquisition

- Cable Television Laboratories, Inc. (CableLabs), 400 Centennial Parkway Louisville CO 80027-1266 www.cablelabs.com/projects/metadata/
- Internet Engineering Task Force (IETF) Secretariat, c/o Corporation for National Research Initiatives, 1895 Preston White Drive, Suite 100, Reston, VA 20191-5434, Phone 703-620-8990, Fax 703-620-9071, Internet: www.ietf.org
- Time Warner Cable, 290 Harbor Drive, Stamford CT 06902, http://twcable.web.aol.com/Pegasus/
- Worldwide Web Consortium, http://www.w3c.org

3 TERMS AND DEFINITIONS

This specification uses the following terms:

Metadata	Metadata is descriptive data associated with a content asset package or file. It may vary in depth from merely identifying the content package title or information to populate an EPG to providing a complete index of different scenes in a movie or providing business rules detailing how the content package may be displayed, copied, or sold. Separate uses for metadata have originated from the studios, distribution networks (Cable, Satellite), down to the CPE (STBs, PVRs).
Asset Distribution Interface	The Asset Distribution Interface is the means by which assets (content as well as metadata describing that content) are transported from a provider to an Asset Management System. In addition, it provides a mechanism by which 'block' updates may be made to previously distributed assets. These block updates include metadata replacement, content replacement ,adding a new child asset to a previously received asset and deleting an asset.
Asset Management System	An Asset Management System is any entity that stores and manages the lifecycle of Assets.
Asset Distribution System	An Asset Distribution System provides transportation of Assets from the premises of the Assets' Provider to the Premises of the MSO. It interfaces to the Asset Management System using ADI.
Package	A Package is a collection of Assets delivered, tracked, and managed as a unit.
Asset	An Asset is the combination of a Content, which is a physical media file such as an MPEG-encoded movie, combined with the necessary metadata required to use the Content for a given application. Assets may, optionally, be composite such that they contain other Assets.
Update	Update refers to the replacement of a component of an Asset, either the Metadata, the Content, or both. Either component may be updated independently, but, if updated, must be replaced in its entirety.

4 ABBREVIATIONS AND ACRONYMS

This specification uses the following abbreviations:

ADI	Asset Distribution Interface
AMS	Asset Management System
ADS	Asset Distribution System
CORBA	Common Object Request Broker Architecture
CPE	Customer Premise Equipment
DTD	Document Type Definition
FTP	File Transfer Protocol
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
JPEG	Joint Photographic Experts Group
MOD	Movies on Demand
MPEG	Motion Picture Expert Group
PVR	Personal Video Recorder
STB	Set-Top-Box
SVOD	Subscription Video on Demand
URL	Uniform Resource Locator
VOD	Video on Demand
WWW	World Wide Web
XML	eXtensible Markup Language

5 DATA TYPES

5.1 Asset

An asset is a container for any object or set of objects that may be required to implement a service, including video, audio, images, application executables, scripts, configuration files, text, fonts, and HTML pages. In addition to the raw content, metadata is also a part of an asset object that describes characteristics of the asset.

Assets may be recursive in nature; that is, they may contain additional assets, which may, themselves, contain additional assets, and so on. Assets are objects that are used to contain arbitrary content for purposes of tracking and control. An application is a type of asset. Delivering an asset that is an application runs that application.

Assets are contained in a single directory in the distribution environment to avoid fragmentation. It is important not to let asset content get separated from metadata.

Assumptions about assets:

- Asset metadata describes the type of content contained in the asset.
- Assets may either contain their content, or a reference to it.
- An asset may have no content.
- Content is any arbitrary set of bits.
- Assets always have a version number.
- Assets may contain zero or more assets.

Metadata is grouped according to its consumer, i.e., what part of the system is going to be processing it. Possible groups are:

- Asset Management System
- Delivery system (like the video server)
- An application (such as MOD)
- Operational Support System
- Business Support System (the billing system)

Assets are uniquely identified by the combination of the Provider_ID and Asset_ID fields within the ADI.DTD. Each Provider assigns these IDs prior to delivery. These identifiers are persistent across both the ADS and the AMS. They serve as the 'agreed to' identifier between the ADS and the AMS so that and Asset may be referenced for subsequent actions such as delete or update.

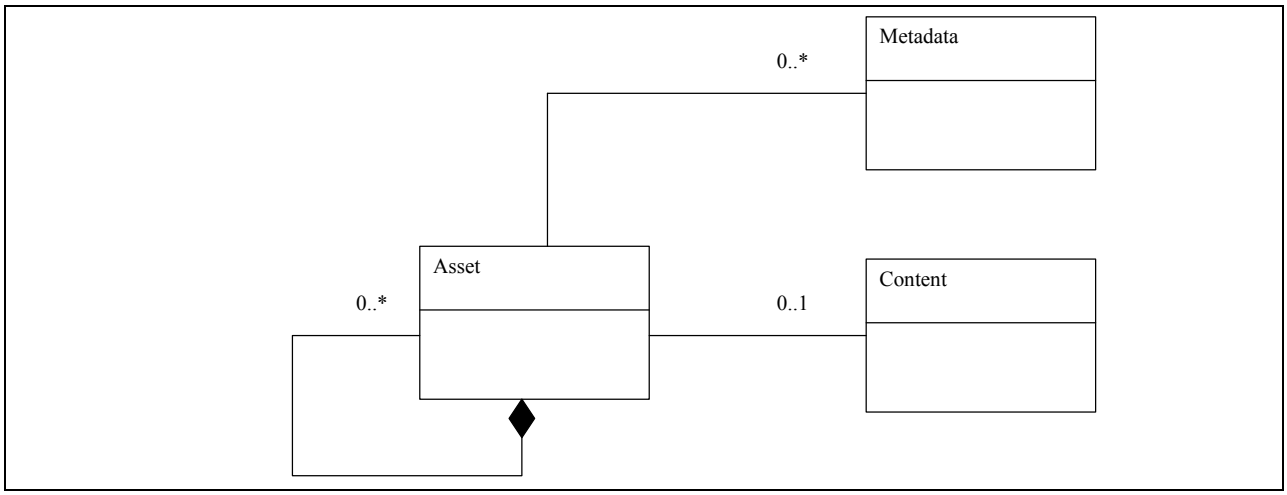

Figure 2 – UML Diagram for Assets

5.2 Metadata

Metadata describes characteristics of an asset, attributes that are inherent in the content of the asset, such as format, duration, size, or encoding method. Values for metadata are determined at the time the asset is created and do not change over time. Metadata is extensible and consists of keyword-value pairs. Asset-specific keywords may be created and values defined at asset creation time.

This data is described in XML so that the parsing routines at the receiving end may be as simple as possible. This also allows for easy extensions with respect to new metadata tag-value pairs. Regardless of the type of metadata, all metadata must be provided, both AMS metadata and application metadata within an assets metadata element.

5.2.1 AMS metadata

Certain metadata is used by the Asset Management System to manage assets. This metadata is common to all assets and is not application specific.

The following metadata elements should be present for all assets delivered to the AMS.

Table 1 – Metadata Elements

Metadata Name	Description	Type
Asset_Name	A string containing the identifying name of the asset. Asset names must be unique within a product.	String
Provider	A unique identifier for the Asset's provider	String
Product	A unique (within the provider's namespace) identifier for the product	String
Version_Minor	An integer representing the minor version number (usually displayed after the decimal point: in Version 7.8, 8 is the minor version number). "*" represents all versions.	String
Version_Major	An integer representing the major version number (usually displayed before the decimal point: in Version 7.8, 7 is the major version number). "*" represents all versions.	String
Description	A human-readable string describing the Asset.	String
Creation_Date	A string representing the date on which the Asset was created, for example: "1999-03-16".	String format yyyy-mm-dd
Provider_ID	A unique identifier for the Asset's provider. The Provider_ID shall be set to the provider's registered internet domain name.	String
Asset_ID	A string containing the identifying name of the asset. An Asset_ID shall uniquely identify an asset within a provider's namespace defined by the Provider_ID attribute. All Asset_ID's will have a maximum length of 20, with the first 4 characters alpha and the last 16 characters numeric	String (alpha/ numeric)
Asset_Class	A string containing the class of the asset, as identified within the content specification that defines the asset. The possible values for this specification are beyond the scope of this specification.	String
Verb	A string containing an action to be performed on the asset. The only valid values for Verb are the empty string (""),) and "DELETE". Note the Verb attribute is optional.	String

5.2.2 Application metadata

An application may specify and use any metadata tags and values that are specific to that application. This document describes the mechanism by which application specific metadata is specified.

Application specific metadata tags and values are not described in this document.

5.3 Content

Content is the actual bits that compose the "payload" of the asset. Content may be an HTML web page, an MPEG transport stream, a JPEG image, or a set top application executable, to name a few examples. Content is any arbitrary collection of bits. There are no restrictions on size or what a content file contains.

6 FILE FORMATS

6.1 ADS Directory Format

6.1.1 Structure

The ADS directory contains the DTD file that describes the XML file syntax, the XML file describing the asset metadata and relationships, and zero or more files containing the content of the assets as shown here:

- XML asset file
- Content File 1
- Content File 2
- …
- Content File N

The DTD file describes the syntax within the XML asset file. The asset file describes the structure of the asset (and its nested assets), all metadata, and contains references to the contents in the same directory. There will always be exactly one DTD file, one asset file, and any number of content files.

6.2 File Formats

6.2.1 XML asset file

The asset metadata and relationships are specified using XML as specified in the AMI DTD (see Appendix I). When performing metadata updates, all application metadata items (App_Data elements) should be provided such that the application metadata items are replaced in their entirety.

6.2.2 Content files

Files containing the content of the assets are in the same directory on the ADS as the XML file. Each content file has a unique name within the directory, and is referenced from within the XML with the “Content Value" item.

If an asset does not contain content, a Content element may be provided with the value attribute containing the literal “NONE”. The absence of a content element will indicate no change to the content.

6.2.3 File Size

There is no maximum file size specified. Therefore, in order to avoid problems with overflowing file pointer sizes on various platforms, it is incumbent on the developers of systems on either side of the Asset Distribution Interface to ensure that they can handle files of arbitrary size. This means that it may be necessary to segment a large file into smaller pieces before, during, or after transmission.

7 MESSAGING PROTOCOL

This document envisions, but does not yet require, a messaging protocol to support the transmission of the ADI files described in section 6. This protocol will provide positive control over the process of distributing Asset Packages from the ADS to the AMS, and will provide positive feedback to the ADS on the success of the distribution, including the success or failure of any file copy operation as well as any additional processing of the Asset involved in installing it into its ultimate repository.

Standardizing the messaging protocol provides for complete interoperability between ADS and AMS vendors, and also lays the groundwork for automatic distribution failure notification and recovery. Failing to standardize on a messaging protocol will result in multiple, per-installation vendor-vendor ADI integrations. This will result in longer deployment times for new content providers. A simplistic approach, such as a simple file copy, would facilitate basic integration, but would limit the ability of content providers to automatically handle failures in Asset packaging.

While not a normative part of this specification, one example of such a messaging protocol be found in Appendix III.

8 EXPECTED BEHAVIORS

While implementation details are not a part of this specification, a messaging protocol implies certain behaviors of each end of the communication. Note that these behaviors are 'logical' functions and do not assume specific behaviors or implementation. These behaviors may be implemented by more than one system.

8.1 Instantiation of a Package

The ADS uses a Package asset to instantiate a set of assets on the AMS. Since a package is a specific type of asset, it is uniquely identified by it's Provider_ID, Asset_ID pair. The package Asset may contain a hierarchy of children assets which are also uniquely identified by the their Provider_ID, Asset_ID pair. The root of this hierarchy is the Package asset. The children assets may contain Application metadata and/or content.

The ADS may expect that an instantiated Asset, which has not been acted upon by the 'delete' action, may be referenced. The reference to an asset shall contain the path through the hierarchy of assets to 'reach' that asset. Note that this does not specify implementation, only that a persistent logical relationship exists among the assets in the package.

8.2 Assignment of a Child Asset to a Parent Asset

If an asset is specified as a child of an asset in the ADI document, that child asset is assigned to that parent asset and added to the asset hierarchy. The initial delivery of a package may contain Assets and their hierarchy. Within the package, this ADI document defines a parent-child relationship between 2 assets. This logical tree structure is persistent until acted upon by a delete command.

A subsequent delivery of the package may contain new Assets which may be defined as children of existing Assets. In this manner it is possible to stagger the delivery of assets. For example, business needs may drive the delivery of a promo video clip prior to the delivery of the feature which it is promoting.

Appendix II.1, 'Initial Pitch of Asset' contains an example of the parent-child assignment within an initial Pitch. Appendix II.3, 'Add an additional Asset', contains an example of the assignment of a new child to an existing parent.

8.3 Assignment of Application Metadata and Content to an Asset

If an asset within an ADI document contains Application Metadata and/or Content, those elements are assigned to that asset. Application Metadata and Content elements are operated upon as a unit, that is, a new element shall completely replace an existing element.

The ADS may expect that an instantiated asset, which has not been acted upon by the 'delete' verb, may be referenced and that the same new/replacement may be used on that asset

Appendix II.2, 'Replace Asset Metadata and/or Content', contains an example of replacement of both Application Metadata and Content.

8.4 Delete an Asset

The Delete verb will immediately remove the specified asset, its metadata, content, and all of its children assets from the AMS. Those Asset(s) will no longer be available to either subscribers or to the ADS. This verb is intended to be used only in an emergency, when immediate action is required (e.g. wrong content was assigned to an asset). It is not necessary for the ADS to specify all children assets that will be operated upon by a 'delete' against a parent asset

Again, note that specific implementation is not intended or implied, however a 'delete' against a package asset will result in the package and all of its child Assets being removed.

Appendix II.4, 'Delete an Asset', contains an example. The delete only applies to the specified asset, not the entire package.

Appendix I – DTD file format

The examples provided are for illustrative purposes and hence do not necessarily reflect a specific content specification. Specifically, these examples do not represent valid CABLELABS VOD 1.0 instances.

```
<!-- DTD for Package-->
<!--CableLabs Asset Distribution Interface version 1.1 -->
<!-- <!ENTITY amp "&#38;#38;"> -->
<!ELEMENT ADI (Metadata, Asset* )>
<!ELEMENT Asset ( Metadata, Asset*,  Content?)>
<!ELEMENT Metadata (AMS, App_Data*)>
<!ELEMENT AMS (#PCDATA)>
<!ATTLIST AMS
       Asset_Name CDATA #REQUIRED
       Asset_ID      CDATA #REQUIRED
       Asset_Class CDATA #REQUIRED
       Provider CDATA #REQUIRED
       Provider_ID CDATA #REQUIRED
       Product CDATA #REQUIRED
       Version_Minor CDATA #REQUIRED
       Version_Major CDATA #REQUIRED
       Description CDATA #REQUIRED
       Creation_Date CDATA #REQUIRED
       Verb CDATA #IMPLIED
>
<!ELEMENT App_Data (#PCDATA)>
<!ATTLIST App_Data
       App CDATA #REQUIRED
       Name CDATA #REQUIRED
       Value CDATA #REQUIRED
>
<!ELEMENT Content (#PCDATA)>
<!ATTLIST Content
       Value CDATA #REQUIRED
>
```

Appendix II – Example ADI XML Documents

II.1 Initial pitch of Assets

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE ADI SYSTEM 'ADI.DTD'>

<ADI>
<Metadata>
   <AMS      Asset_Name="CaptainCorellisMandolinpackage"
Provider="iN DEMAND"
Product="MOD" Version_Major="1" Version_Minor="0"
Description="Captain Corellis Mandolin asset package"
Creation_Date="2002-03-05"
Provider_ID="indemand.com"
Asset_ID="UNVA2001081701003000"
Asset_Class=”package” />
</Metadata>

<Asset>
<Metadata>
      <AMS Asset_Name="Captain Corellis Mandolin title"
Provider="iN DEMAND" Product="MOD" Version_Major="1" Version_Minor="0"
Description="Captain Corellis Mandolin title asset"
Creation_Date="2002-03-05"
Provider_ID="indemand.com"
Asset_ID="UNVA2001081701003001"
Asset_Class="title" />

<App_Data App="MOD" Name="Title"
Value="Captain Corellis Mandolin" />
      <App_Data App="MOD" Name="Summary_Short" Value="On a Greek island
during World War II, a young woman falls for an invading Italian
soldier while her lover is away fighting for the Greek army. Nicolas
Cage, Penelope Cruz. Directed by John Madden. (New Release, 2001,
Romantic Drama)" />
      <App_Data App="MOD" Name="Rating" Value="R" />
      <App_Data App="MOD" Name="Display_Run_Time" Value="02:08" />
      <App_Data App="MOD" Name="Run_Time" Value="02:08:00" />
      <App_Data App="MOD" Name="Year" Value="2001" />
      <App_Data App="MOD" Name="Actors" Value="Nicholas,Cage” />
      <App_Date App=”MOD” Name="Actors" Value="Cruz,Penelope” />
      <App_Date App=”MOD” Name="Actors" Value="Bale,Christian” />
      <App_Data App=”MOD” Name=”Actors” Value=”Hurt,John” />
      <App_Data App="MOD" Name="Director" Value="Madden,John" />
      <App_Data App="MOD" Name="Studio”_Value="Universal Studios" />
      <App_Data App="MOD" Name="Genre" Value="Drama" />
      <App_Data App=”MOD” Name=”Category” Value=” New Releases/Drama,
New Releases/Movies A to Z” />
      <App_Data App="MOD" Name="Provider_Asset_ID" Value="40600" />
      <App_Data App="MOD" Name="Licensing_Window_Start" Value="22-03-2002" />
      <App_Data App="MOD" Name="Licensing_Window_End" Value="05-07-2002" />
      <App_Data App="MOD" Name="Contract_Name" Value="Contract Name" />
      <App_Data App="MOD" Name="Royalty_Percent" Value="0" />
      <App_Data App="MOD" Name="Royalty_Minimum" Value="0" />
      <App_Data App="MOD" Name="Royalty_Flat_Rate" Value="0" />
      <App_Data App="MOD" Name="Box_Office" Value="26" />
      <App_Data App="MOD" Name="Suggested_Price" Value="3.95" />
      <App_Data App="MOD" Name="Maximum_Viewing_Length" Value="24:00:00" />

</Metadata>

<Asset>
```

```
      <Metadata>
<AMS Asset_Name="Captain Corellis Mandolin feature"
Provider="iN DEMAND"
Product="MOD" Version_Major="1" Version_Minor="0" Description="Captain Corellis
Mandolin feature asset" Creation_Date="2002-03-05"
Provider_ID="indemand.com"
Asset_ID="UNVA2001081701003002"
Asset_Class="movie" />
      </Metadata>
      <Content Value="Mandolin.mpg" />
   </Asset>

<Asset>
      <Metadata>
<AMS Asset_Name="Captain Corellis Mandolin trailer"
Provider="iN DEMAND"
Product="MOD" Version_Major="1" Version_Minor="0" Description="Captain Corellis
Mandolin trailer asset" Creation_Date="2002-03-05"
Provider_ID="indemand.com"
Asset_ID="UNVA2001081701003003"
Asset_Class="preview" />
        <App_Data App="MOD" Name="Rating_Type" Value="MPAA" />
        <App_Data App="MOD" Name="Rating" Value="R" />
        <App_Data App="MOD" Name="Run_Time" Value="00:01:30" />
  </Metadata>
      <Content Value="MandolinTR.mpg" />
    </Asset>

   <Asset>
      <Metadata>
        <AMS Asset_Name="Captain Corellis Mandolin artwork"
Provider="iN DEMAND"
Product="MOD" Version_Major="1" Version_Minor="0" Description="Captain Corellis
Mandolin artwork asset"
Creation_Date="2002-03-05"
Provider_ID="indemand.com"
Asset_ID="UNVA2001081701003004"
Asset_Class="poster"/>
      </Metadata>
      <Content Value="captaincorellis.bmp" />
    </Asset>
</Asset>
</ADI>
```

II.2 Replace the Asset Metadata

The structure for metadata replacement is to nest the asset being modified within its parent asset.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE ADI SYSTEM 'ADI.DTD'>
<ADI>
<Metadata>
   <AMS      Asset_Name="CaptainCorellisMandolinpackage"
Provider="iN DEMAND"
Product="MOD" Version_Major="1" Version_Minor="0"
Description="Captain Corellis Mandolin asset package"
Creation_Date="2002-03-05"
Provider_ID="indemand.com"
Asset_ID="UNVA2001081701003000"
Asset_Class=”package” />

</Metadata>

<Asset>
```

```
<Metadata>
      <AMS Asset_Name="Captain Corellis Mandolin title"
Provider="iN DEMAND" Product="MOD" Version_Major="1" Version_Minor="0"
Description="Captain Corellis Mandolin title asset"
Creation_Date="2002-03-05"
Provider_ID="indemand.com"
Asset_ID="UNVA2001081701003001"
Asset_Class="title" />
</Metadata>

<Asset>
      <Metadata>
<AMS Asset_Name="Captain Corellis Mandolin trailer"
Provider="iN DEMAND"
Product="MOD" Version_Major="1" Version_Minor="0" Description="Captain Corellis
Mandolin trailer asset" Creation_Date="2002-03-10"
Provider_ID="indemand.com"
Asset_ID="UNVA2001081701003003"
Asset_Class="preview" />
        <App_Data App="MOD" Name="Rating_Type" Value="MPAA" />
        <App_Data App="MOD" Name="Rating" Value="G" />
        <App_Data App="MOD" Name="Run_Time" Value="00:01:29" />
  </Metadata>
</Asset>
</Asset><!-- end of title asset -->
</ADI>
```

II.3 Add an additional Asset

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE ADI SYSTEM 'ADI.DTD'>

<ADI>
<Metadata>
<AMS   Asset_Name="CaptainCorellisMandolinpackage"
Provider="iN DEMAND"
Product="MOD" Version_Major="1" Version_Minor="0"
Description="Captain Corellis Mandolin asset package"
Creation_Date="2002-03-05"
Provider_ID="indemand.com"
Asset_ID="UNVA2001081701003000"
Asset_Class=”package” />
</Metadata>
<Asset>
<Metadata>
      <AMS Asset_Name="Captain Corellis Mandolin title"
Provider="iN DEMAND" Product="MOD" Version_Major="1" Version_Minor="0"
Description="Captain Corellis Mandolin title asset"
Creation_Date="2002-03-05"
Provider_ID="indemand.com"
Asset_ID="UNVA2001081701003001"
Asset_Class="title" />
</Metadata>
   <Asset>
      <Metadata>
        <AMS  Asset_Name="Captain Corellis Mandolin artwork2"
Provider="iN DEMAND"
Product="MOD" Version_Major="1" Version_Minor="0" Description="Captain Corellis
Mandolin artwork asset2"
Creation_Date="2002-11-03"
Provider_ID="indemand.com"
Asset_ID="UNVA2001081701003005"
Asset_Class="poster"
/>
        <App_Data App="MOD" Name="Asset_Type" Value="poster" />
      </Metadata>
      <Content Value="captaincorellis2.bmp" />
    </Asset>
    </Asset>
</ADI>
```

II.4 Delete an Asset

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE ADI SYSTEM 'ADI.DTD'>

<ADI>
<Metadata>
   <AMS      Asset_Name="CaptainCorellisMandolinpackage"
Provider="iN DEMAND"
Product="MOD" Version_Major="1" Version_Minor="0"
Description="Captain Corellis Mandolin asset package"
Creation_Date="2002-05-03"
Provider_ID="indemand.com"
Asset_ID="UNVA2001081701003000"
Asset_Class=”package” />
</Metadata>
<Asset>
<Metadata>
      <AMS Asset_Name="Captain Corellis Mandolin title"
```

```
Provider="iN DEMAND" Product="MOD" Version_Major="1" Version_Minor="0"
Description="Captain Corellis Mandolin title asset"
Creation_Date="2002-03-05"
Provider_ID="indemand.com"
Asset_ID="UNVA2001081701003001"
Asset_Class="title" />
</Metadata>
<Asset>
      <Metadata>
<AMS Asset_Name="Captain Corellis Mandolin feature"
Provider="iN DEMAND"
Product="MOD" Version_Major="1" Version_Minor="0" Description="Captain Corellis
Mandolin feature asset" Creation_Date="2002-03-05"
Provider_ID="indemand.com"
Asset_ID="UNVA2001081701003002"
Asset_Class="movie"
Verb="DELETE" />
      </Metadata>
   </Asset>
</Asset><!-- end of title asset -->
</ADI>
```

Appendix III – Reference Messaging Protocol Implementation

This appendix captures a reference implementation of ADI which utilizes a subset of the Interactive Services Architecture (ISA), which is a specification based on CORBA. This appendix describes the subset of ISA relating to the management of Package objects, which is a function of the ADS. The AMS implements the Package object and PackageFactory objects. The ADS creates and provisions the Package object by making CORBA calls on the AMS. The ADS transfers a package of assets by asking the PackageFactory object to create a Package object, and then provisioning that Package object with the information needed for the AMS to transfer all of the assets contained in it from the ADS to the appropriate locations.

III.1 Locating the PackageFactory

The ADS must first locate the PackageFactory object, which is responsible for creating and managing Packages in the system. This is done by calling the **`resolve()`** function of the Naming Service. This is expected to be done once when the ADS is first initialized.

```
Object resolve (in Name)
```

This returns an object reference to a Package Factory that will be used to locate and or create packages.

III.2 Locating a Package

The ADS should first determine if the Package to be processed already exists on the AMS. The ADS will determine if the Package already exists by calling the **`find()`** function on the PackageFactory**.**

```
ServantBase find (in Name)
```

The ADS should provide the Provider_ID and Asset_ID as the name. Specifically, the format of the name provided should be a concatenation of the Provider_ID, '/' and the Asset_ID taken from the AMS metadata element of the XML instance document.

This returns an object reference to the Package that will be provisioned if the Package already exists.

III.3 Creating a Package

If the result of the ADS performing 6.2.1 fails to result in a Package object reference, the ADS should create a Package object to be provisioned.

Using the PackageFactory object located from the naming service, the ADS should use the **`createServan`**t() function on the PackageFactory to instantiate a Package object. An object reference to the newly created Package object is returned.

```
ServantBase createServant (in string name)
```

The ADS should provide the Provider_ID and Asset_ID as the name. Specifically, the format of the name provided should be a concatenation of the Provider_ID, '/' and the Asset_ID taken from the AMS metadata element of the XML instance document.

This returns an object reference to the Package that will be provisioned.

III.4 Provisioning the Package

In ISA, a newly created object is in an unprovisioned state, and is considered to be out of service. This means the Package object must be provisioned with the package information in the ADS. This is done by calling the `provision` function on the Package object. The arguments passed to this function should be the URL of the XML file on the ADS, and a value that sets the state of the object to in service.

```
void provision (in AdministrativeState theAdministrativeState,  in
string theUrl)
```

The URL passed in the `provision` function should contain the protocol that the AMS may use to copy the files from the ADS, the authentication credentials (login and password) and a hostname for the ADS that can be resolved by the AMS, and the path and file name of the XML asset file.

The `provision` message will cause the content files to be fetched from the ADS as described below. Completion of the `provision` function will not occur until all the content files have been transferred.

During processing of the `provision` message, the AMS may encounter an error. In such an event, the AMS will provide an Exception which indicates the source of the failure. The following exceptions should be provided:

XMLProcessingException

An invalid XML document was provided.

TransferException

A TransferException will contain a code indicating one of the following conditions: NotEnoughSpace, CheckSumMismatch, SizeMismatch, ConnectionRefused, NetworkTimeout, NoRoute, HostnameLookup.

VersionException

An update to a package and or asset could not be processed due to a version disparity.

All other failures should result in a ProvisioningException or InvalidStateException.

III.5 Transferring content

The protocol in the URL of the XML file determines the protocol that the AMS will use to fetch the XML file from the ADS unless otherwise indicated by the content file URL. If the content file location URL is simply a filename, the content files are assumed to be in the same relative location as the XML file. The ADS SHALL provide for one of the following protocols:

- ftp
- http
- file

Specifically, the ADS should minimally support the “get”, “mode” operations for ftp.

The AMS should fetch the XML file, parse it, and then should fetch the content files described within it. When all of the content has been transferred, or an error is encountered, the `provision` function will return.

III.6 ISA Package Distribution Event Trace Diagrams

Figure 3, 'Initial Package Propagation – ADS view', illustrates an initial package propagation scenario for package propagation. Figure 4, 'Package Update Propagation – ADS view', illustrates an update to package propagation scenario.

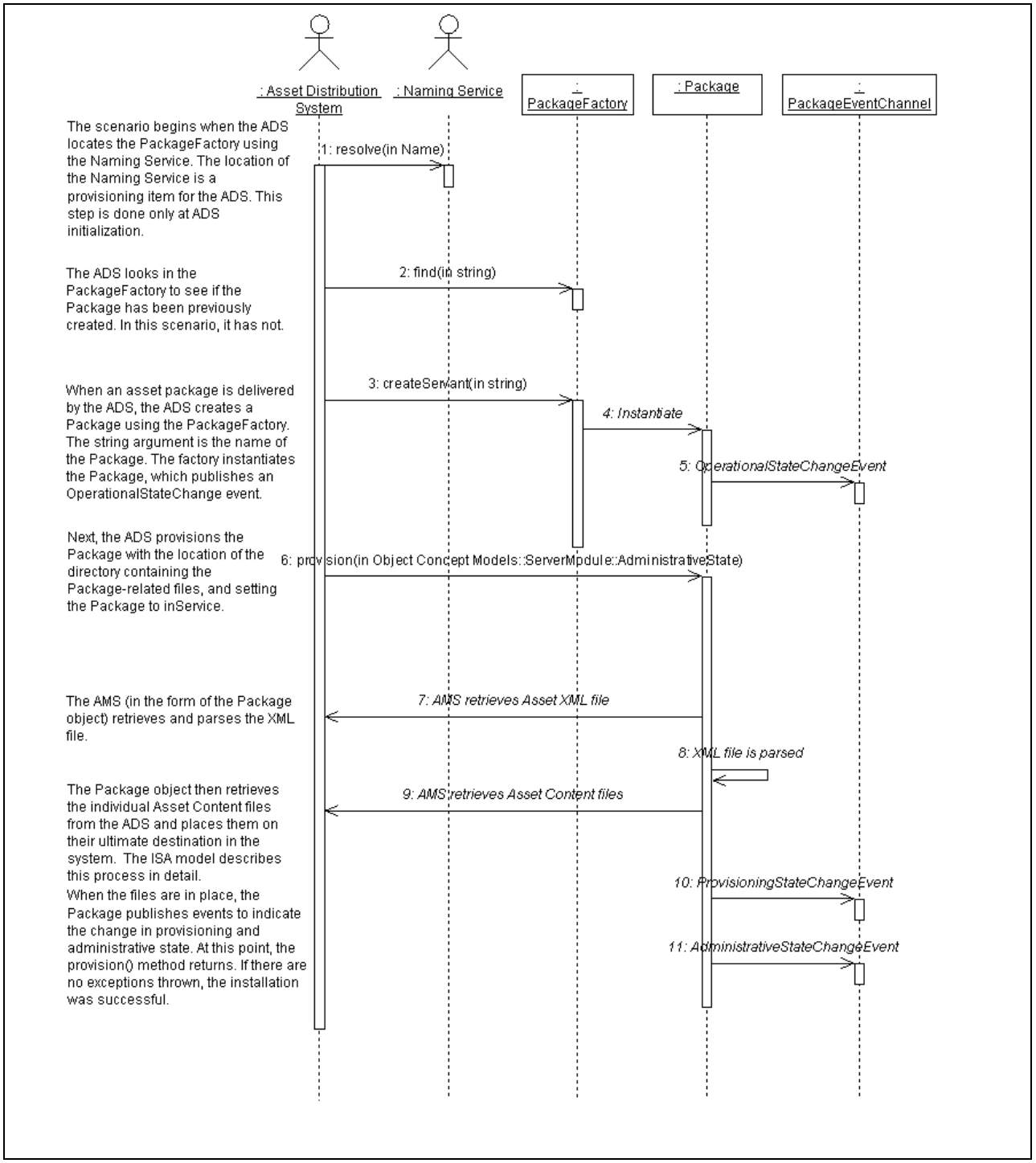

Figure 3 – Initial Package Propagation - ADS view

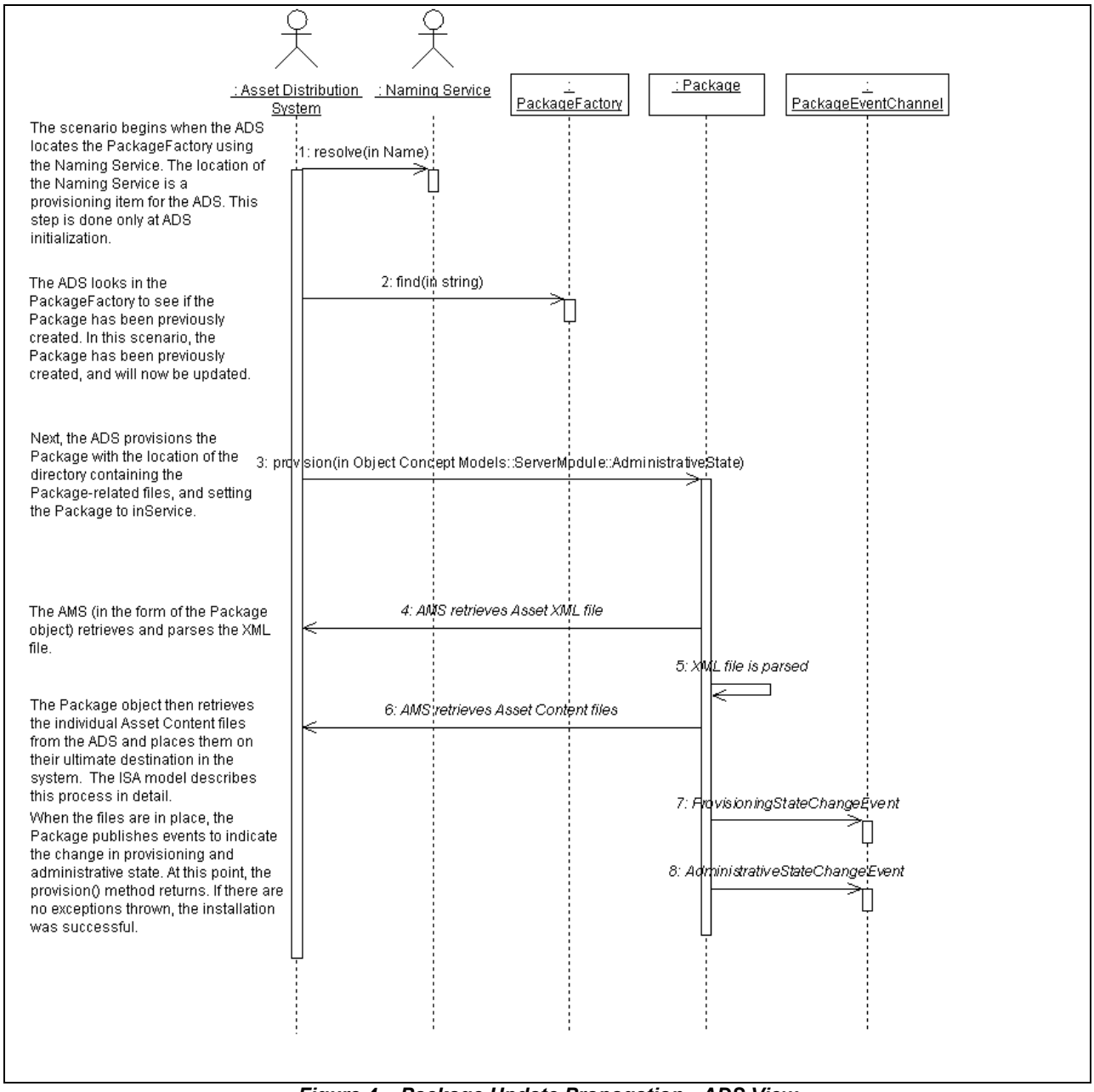

Figure 4 – Package Update Propagation - ADS View

```
//Source file: c:/isaIdl/PackageComponent.idl

#ifndef __PACKAGECOMPONENT_DEFINED
#define __PACKAGECOMPONENT_DEFINED

/* CmIdentification
  %X% %Q% %Z% %W% */

/*
 * Copyright 2000, 2001, 2002 Time Warner Cable. All rights reserved.
 *
 * ISA version 1.4 PRELIMINARY 2
 */

#include "AssetComponent.idl"
#include "ServerComponent.idl"
#include "ProductComponent.idl"
#include "ContentComponent.idl"
#include "CosEventChannelAdmin.idl"

#pragma prefix "isa.twc.com"



/* The Pegasus Asset Distribution Interface (ADI) defines a Package to be a
specialization of the Asset. In ISA, a Package is a derived type of Asset. This module
defines the relationships and interfaces for the Package component. */

module PackageModule {
       interface Package;


       /* An event channel to publish Package-related events. */

       interface PackageEventChannel : CosEventChannelAdmin::EventChannel {
       };

       exception VersionException {
              string message;
              ServerModule::CompletionCode theCompletionCode;
       };

       exception XmlProcessingException {
              string message;
              ServerModule::CompletionCode theCompletionCode;
       };

       enum TransferExceptionCode {

              tec_NotEnoughSpace,
              tec_CheckSumMismatch,
              tec_SizeMismatch,
              tec_ConnectionRefused,
              tec_NetworkTimeout,
              tec_NoRoute,
              tec_HostNameLookup
       };

       exception TransferException {
              string message;
              ServerModule::CompletionCode theCompletionCode;
              TransferExceptionCode theTransferExceptionCode;
       };

       /* Manages the lifecycle of Package objects. */
```

```
	interface PackageFactory : AssetModule::AssetFactory {
		typedef sequence <PackageModule::Package> aPackage_def;

		attribute aPackage_def aPackage;
	};

	/* The Package maintains the relationship between all the Assets in a given
distribution. In some ways it is similar to a bill of lading. But it has the
additional responsibility of managing the process of installing the assets delivered
in the Package. This requires it to parse XML metadata and allocate ISA metadata
objects, and to interact with Application objects and ContentStore and Content objects
to provide for the storage of the Assets.

	The Package metadata is specified in the ADI spec. The content for a Package
asset is defined to be the install script for the package. */

	interface Package : AssetModule::Asset {
		typedef sequence <ProductModule::Product> aProduct_def;

		/* theDirectoryUrl is the location on the Asset Distribution System of
the directory that contains the ADI metadata and all the individual content media
files. */

		attribute string theDirectoryUrl;
		attribute PackageFactory thePackageFactory;
		attribute aProduct_def aProduct;

		/* For the Package, the provision operation not only provides for
configuration of the object, but also is the instruction to the Package to begin its
function of installing the Assets that are collateral with the Package. When the
Provision operation returns, all Assets are installed in their final repositories.
Provision should raise exceptions if there is any failure in the installs. Presently
the entire model needs identification and definition of a great many exceptions.
		@roseuid 39049CAF0050 */
		void provision (
			in ServerModule::AdministrativeState theAdministrativeState,
			in string theUrl
			)
			raises
(ServerModule::ProvisioningFailed,ServerModule::UnspecifiedException,ServerModule::Inv
alidStateChange,PackageModule::TransferException,PackageModule::VersionException,Packa
geModule::XmlProcessingException);

		/* getProvisioning allows the provisioning of a Package to be retrieved.
Package Provisioning is not extremely interesting, in general.
		@roseuid 394E86DA035C */
		void getProvisioning (
			out string name,
			out ProductModule::ProductList Products,
			out MetadataModule::MetadataList theMetadataList,
			out ContentModule::ContentList theContents,
			out AssetModule::AssetList ChildAssets,
			out AssetModule::Asset theParentAsset,
			out string theDirectoryUrl,
			out ServerModule::AdministrativeState theAdministrativeState,
			out ServerModule::OperationalState theOperationalState
			)
			raises
(ServerModule::ServantNotProvisioned,ServerModule::UnspecifiedException);

	};

};

#endif
```

III.7 Provision Functions

The possible exceptions thrown by the `provision` function of the Package object are:

Table 2 – Provision Functions

Exception	Description
XMLProcessingException	An invalid XML document was provided.
TransferException	A TransferException will contain a code indicating one of the following conditions: NotEnoughSpace, CheckSumMismatch, SizeMismatch, ConnectionRefused, NetworkTimeout, NoRoute, HostnameLookup.
VersionException	An update to a package and or asset could not be processed due to a version disparity.
ProvisioningFailed	Generalized failure.
InvalidStateException	Package could not be placed into service.
UnspecifiedException	Other failure, not stated in this specification.