Adobe Content Server Packaging Guide

This document is a complement to ACS documentations. We are often asked how to start developing one's own packaging tool for automating the workflow. This document does not cove in-depth technical implementation, but rather it is aimed to give developers and ACS operators a guide to help you get started.

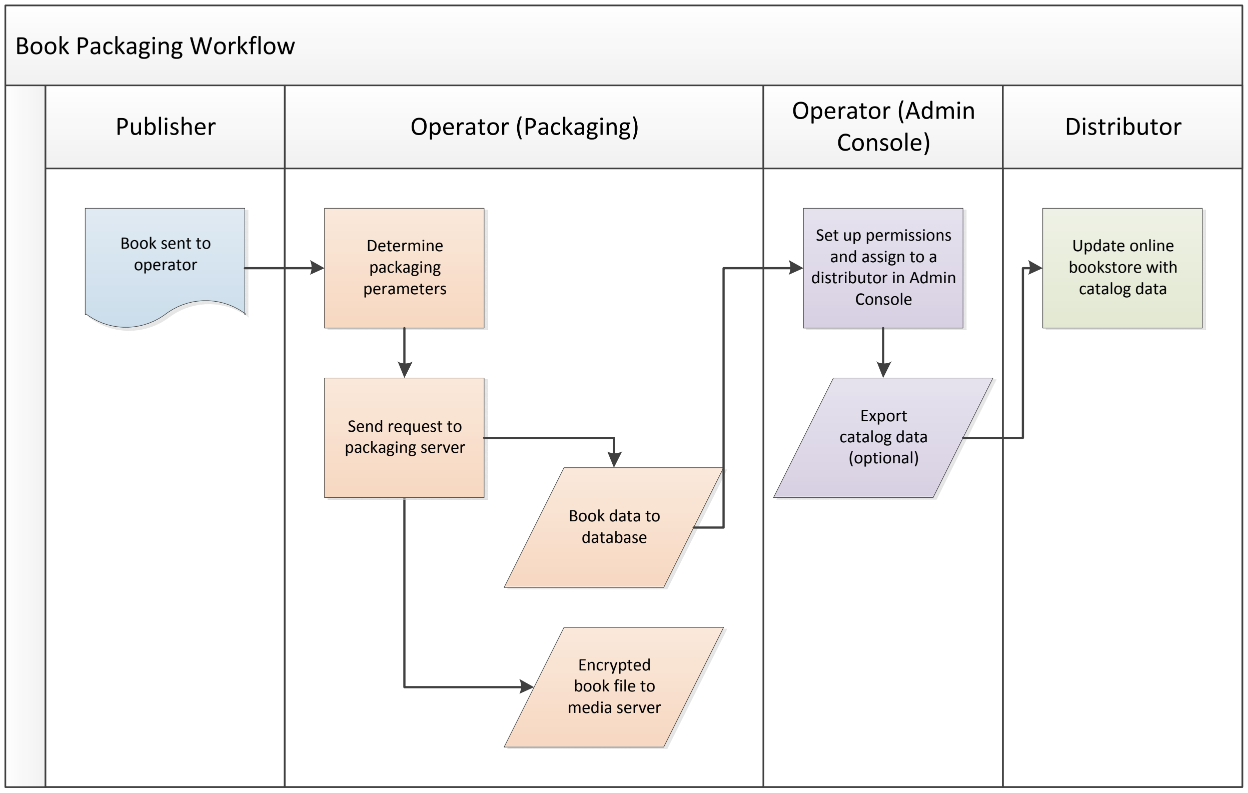
Feedback and Questions, contact Dan Strauss and Ching Yue.
Last Update: March 29, 2012.

Table of Contents

1 Overview

There are 3 major players involved in hosting Adobe Content Server: publishers (providing PDF and EPUB files), operators (running ACS software and services), and distributors (running bookstores and libraries). This document assumes that the publisher is separate from the operator and distributor and will focus on the activities these last two entities must perform.

The following diagram depicts a general workflow for the packaging process.


Book Packaging Workflow
Publisher
Operator (Packaging)
Operator (Admin Console)
Distributor
Book sent to operator
Determine packaging perameters
Send request to packaging server
Book data to database
Encrypted book file to media server
Set up permissions and assign to a distributor in Admin Console
Export catalog data (optional)
Update online bookstore with catalog data

By definition, an operator’s role is to host the fulfillment center for ACS and to handle the steps to prepare books for fulfillment. Publishers provide the source books and these books are packaged into encrypted books with DRM protection.

The distributors are assigned these books and also control some aspects of the book permissions. The distributors are the bookstores and/or libraries.

An operator often works with multiple publishers and, in many cases, multiple distributors as well. It is also possible that a publisher and a distributor work with more than one operator. Since the operator hosts the ACS server (think of it as a hub to gather and disseminate books), the services are provided by the operator outward to publishers and distributors.

On a higher level, the first few steps a book goes through are:

1. The publisher decides a book is ready and determines the base permissions for the book.
2. The operator or publisher packages the book and uploads the book to a media server.
3. The operator assigns the book to a distributor
4. The operator assigns further permissions to the book based on the particular distributor
5. The book becomes available in the distributor’s catalog

The current system is designed around the operator role. The publisher and distributors have very little interaction with the system by current design. This document will cover how you, the operator, can package the book and prepare it for display in the online catalog.

2 Using the Included Tools

The easiest and most basic way to package a single book is to use the Java tool included with Adobe Content Server called “Upload Test” (located in UploadTest-1_2.zip). The tool is a JAR file that can be run from the command line to package a book while optionally adding permissions, metadata, and a thumbnail image.

Please note that it is best practice to manually type out all commands and command arguments. Copying and pasting from other documents (this one included) may lead to strange characters getting into the command, causing errors to occur. Also note that any time a file name is mentioned with numbers or dates in its name, it is subject to change as newer versions are released. However, the base names of these files should remain the same.

2.1 Basic Packaging Command Line

Reference: ACS Quick Start Guide Section 3 "Packaging Books" and User Manual Section 11 "Packaging Books with Content Server"

The basic format of a command is as follows:

```
java –Xmx1024M -jar UploadTest-1_2.jar http://[YOUR
WEBSITE]/packaging/Package [PDF OR EPUB FILE] -pass [PASSWORD]
```

The "`-Xmx`" argument specifies the heap size. At least 512 MB is recommended, with more if the book to be packaged is large. `[YOUR WEBSITE]` is the URL and port number where you have ACS installed. `[PDF OR EPUB FILE]` is the path and name of the book you wish to package. `[PASSWORD]` is the password you use to log in to the Admin Console.

To confirm that your packaging server should be working, use the statuscheck as following from a web browser. This doesn't guarantee that everything is working, but it is a very good indicator whether the system is running or not.

```
http://[YOUR WEBSITE]/packaging/statuscheck
```

2.2 Including Extra Files in Packaging

In addition to the arguments you are required to provide, there are optional arguments as well. The two that you will probably use most often include a thumbnail image file and an XML file in the packaging.

To include an image file, use the image's file extension as an argument. The options are "`-png`", "`-jpg`", "`-jpeg`", and "`-gif`". When this option is used, the packaging tool will search for an image file with that extension matching the name of the book. It will take the path and name of the book file, remove the .epub or .pdf extension, and replace it with the image extension. If a file with that name is not found, the process will still complete, packaging the book without a thumbnail, though it will print a message saying that it could not find the image.

Similarly, you can include an XML file. Passing the "`-xml`" argument will make the packaging tool search for an XML file in the same way it searches for an image file. The XML file should contain metadata, permission information, or both. Example XML files are included with ACS in sample_books.zip and are a good general resource. For more information on the specifics of formatting this file, refer to section 11.2 (Packaging Tool Example) in the User Manual.

2.3 Using Admin Console to Finish Packaging

Reference: ACS Quick Start Guide Section 4 "Admin Console" and User Manual Section 14 "Admin Console"

Once all the parameters are given to the tool, the book will be packaged. The book will be encrypted with Adobe DRM and it will be added to the database along with its permissions, metadata, and thumbnail. Also, a copy of the encrypted book and its thumbnail will be copied to the media server location specified in your packaging configuration file.

After the book has been packaged, however, there are still a few steps that need to be taken before it's ready for your e-commerce site. Primarily, a newly packaged book needs to be added to a distributor and be assigned distributor specific permissions (distribution rights).

2.3.1 Add Distributor

You need to add at least one distributor in Admin Console. To do this, use the Distributor's Panel, filling out all relevant information for the distributor such as name, URL, and link expiration time.

2.3.2 Assign Distribution Rights

The next step would be to assign distributor rights to the book. Any permissions set during packaging will only be applied as "base permissions". If you wish for a particular distributor to have its own unique set of restrictions, or if you wish to make a book loanable, you will have to assign these permissions after the book has already been packaged. For setting loaned books, search for "ACS loan" in our knowledgebase at http://kb.datalogics.com/knowledgeHome.

This step can be done through the Admin Console or a tool of your own, but not through the Upload Test tool.

2.3.3 Export Book Catalog

Finally, once the book is packaged and assigned to a distributor with the appropriate permissions, the changes to the distributor's catalog needs to be reflected on your website. From the Admin Console, you can export a catalog XML containing the metadata, resource locations, and other information for each book assigned to a particular distributor. The sample store uses this XML to populate the catalog, so sample code for utilizing that file is available.

See the QueryTool section below for alternative method of extracting resources.

2.4 Using QueryTool

2.4.1 Purpose and Parameters

The QueryTool sends a QueryResourceItems request to the admin console server. QueryResourceItems retrieves a list of resourceitems in the format that can be used for the bookstore.

The output of this command is very similar to the books.xml file exported from Admin Console.

URL to use for this command is: http://SERVER_NAME/admin/QueryResourceItems.

This URL must be the first command line argument.

This tool also accepts a number of command-line flags that extend its functionality.

Table 1. QueryTool Command Line Options

Options	Purpose
-d distributorUUID	Specify a user-defined distributor UUID to make the request. This flag must be followed by the UUID to use. Remember to use the correct password.
-f or –file filename	Specify that the server response (resourceItemList) should be written to an XML file (for use with extracting resource UUIDs for DistribTool). This flag must be followed by the name (or path/name) of the output file to use.
-p or –pass text-password	Text password set for the distributor. For built-in distributor, this is the password used to login to the Admin Console. The request HMAC sent in the QueryTool will be signed with this password. (The tool will take the raw SHA1 hash of the password to use as HMAC Key Bytes). This flag must be followed by the password text to use.
-p64 or -pass64 SharedSecret	Base64-encoded Shared Secret for the distributor. The tool will decode the password string and use the resulting bytes as HMAC Key Bytes. This flag must be followed by the Base64-encoded password String to use.
-v or -verbose	Enable verbose display mode. For best functionality, use this flag FIRST if you are using multiple flags.
-rf or –resourceFile filename	Extract the resource UUIDs from the server response and output them to a text file (for use as a resourceFile for DistribTool). This flag must be followed by the name (or path/name) of the output file to use .

2.4.2 Query a list of resources from the built-in distributor

```
java -jar C:\ACS\Adobe_ContentServer_4_1_2\UploadTest-1_2\QueryTool-1_2.jar
http://acs.server.com:8080/admin/QueryResourceItems -p admin
```

For built-in distributor, it is much easier to know the text password, so in this example, -p option is used.

2.4.3 Query a list of resources from a specific distributor

```
java -jar C:\ACS\Adobe_ContentServer_4_1_2\UploadTest-1_2\QueryTool-1_2.jar
http:// acs.server.com:8080/admin/QueryResourceItems -d urn:uuid:bfd6065e-
05e7-4d4b-a5fc-ae171cca44f0 -p64 HpNFJ1Dhs/NdA3QCIFRGQCqVJ40= -f C:\
Packaging\QueryToolOutput-Store.xml -rf c:\Packaging\uuidlist.txt
```

This queries a list of resources from a specific distributor. It is much easier to find out the shared secret from Admin Console. So in this example, -p64 is used.

Note that in this example, we are also writing the output to two files, one is to hold the xml output of the <resourceItemList> and the 2nd is to hold the list of resource UUIDs. The resource UUID list can be the input to DistribTool.

2.4.4 Sample Output

The list of resources returned from querying the built-in distributor:

```
<resourceItemList xmlns="http://ns.adobe.com/adept">
  <resourceItemInfo>
    <resource>urn:uuid:28a1b8d8-a331-49e5-b35f-b4c00f66f635</resource>
    <resourceItem>0</resourceItem>
    <metadata>
      <dc:title xmlns:dc="http://purl.org/dc/elements/1.1/"></dc:title>
      <dc:creator xmlns:dc="http://purl.org/dc/elements/1.1/"></dc:creator>
      <dc:format
xmlns:dc="http://purl.org/dc/elements/1.1/">application/pdf</dc:format>
    </metadata>
    <src>http://acs.server.com:8080/media/28a1b8d8-a331-49e5-b35f-
b4c00f66f635.pdf</src>
    <downloadType>simple</downloadType>
    <licenseToken>
      <resource>urn:uuid:28a1b8d8-a331-49e5-b35f-b4c00f66f635</resource>
    </licenseToken>
  </resourceItemInfo>
  More <resourceItemInfo>…</resourceItemInfo> repeated
</resourceItemList>
```

The list of resources returned from querying a specific distributor:

```
<resourceItemList xmlns="http://ns.adobe.com/adept">
  <resourceItemInfo>
    <resource>urn:uuid:d07cd8da-ab8c-491d-87da-b63735712dd3</resource>
    <resourceItem>0</resourceItem>
    <metadata>
      <dc:title xmlns:dc="http://purl.org/dc/elements/1.1/">A Tale of Two
Cities</dc:title>
      <dc:creator xmlns:dc="http://purl.org/dc/elements/1.1/">Dickens,
Charles</dc:creator>
```

```
      <dc:format
xmlns:dc="http://purl.org/dc/elements/1.1/">application/epub+zip</dc:format>
      <dc:publisher xmlns:dc="http://purl.org/dc/elements/1.1/">FeedBooks
(www.feedbooks.com)</dc:publisher>
      <dc:language
xmlns:dc="http://purl.org/dc/elements/1.1/">en</dc:language>
      <dc:description
xmlns:dc="http://purl.org/dc/elements/1.1/"></dc:description>
      <dc:identifier
xmlns:dc="http://purl.org/dc/elements/1.1/">urn:uuid:8ec07740-24be-11dd-83f3-
001cc049a027</dc:identifier>
    </metadata>
    <src>http://acs.server.com:8080/media/d07cd8da-ab8c-491d-87da-
b63735712dd3.epub</src>
    <downloadType>simple</downloadType>
    <licenseToken>
      <resource>urn:uuid:d07cd8da-ab8c-491d-87da-b63735712dd3</resource>
      <permissions>
        <display/>
        <excerpt/>
        <print/>
        <play/>
      </permissions>
    </licenseToken>
  </resourceItemInfo>
  More <resourceItemInfo>…</resourceItemInfo> repeated, but only books
assigned to the specified distributor are included here.
</resourceItemList>
```

Note that in this output, you will see the permissions since now the book is queries under a specific distributor. Also only books assigned to the specified distributor are included here.

The list of UUIDs output from –rf flag, in this example, only 2 books are assigned to the given distributor:

urn:uuid:d07cd8da-ab8c-491d-87da-b63735712dd3
urn:uuid:e67d2584-a339-470c-822c-307b7e6c1c6b

2.5 Using DistribTool

2.5.1 Purpose and Input

The DistribTool sends a ManageDistributionRights request to the admin service in order to assign books to a distributor. It does so by creating an XML-based request that is detailed in-depth below.

URL to use: http://YOUR_SERVER/admin/ManageDistributionRights. This URL is specified in the XML configuration file, described below.

There are two external resources that are required for this tool to run. The first is the XML Configuration file, whose path is specified as the first command line argument to running this tool.

The second required external resource for this tool to run is the resourceFile. This file contains a list of resource UUIDs. If this sounds familiar to you, it is because this file is generated by the QueryTool that we covered previously.

In order to run this tool, you must call it with the path to the XML configuration file as the first command-line argument. Following this first argument, you can use any of the recognizable flags. A sample command looks like:

```
java -jar c:\Data\Projects\ACS\Adobe_ContentServer_4_1_2\UploadTest-
1_2\DistribTool-1_2.jar C:\Data\Doc\Packaging\distributool.xml -p admin –
verbose
```

Table 2. DistribTool Command Line Options

Options	Purpose
-p or –pass text-password	Text password set for the distributor. For built-in distributor, this is the password used to login to the Admin Console. The request HMAC sent in the QueryTool will be signed with this password. (The tool will take the raw SHA1 hash of the password to use as HMAC Key Bytes). This flag must be followed by the password text to use.
-verbose	Displays content of package request and detailed server response
-? or -help	Displays the help message

2.5.2 XML Configuration File

The structure of the XML configuration file is detailed below, and a sample is also shown. All of the elements of the XML Configuration except those labeled optional are required.

Table 3. DistribTool XML Configuration File Elements

Options	Purpose
request	Root element of the request. Includes the Adept namespace (xmlns="http://ns.adobe.com/adept") as the default namespace.
serverURL	Contains the full URL to the admin service. The server URL must point to the ManageDistributionRights API. (http://YOUR_SERVER/admin/ManageDistributionRights)
distributionRights	Empty element that contains child elements: distributor, resourceFile, and distributionType. The XML Configuration file must contain one or more distributionRights elements

distributor	the UUID of the distributor to whom the resources specified by resourceFile will be assigned.
resourceFile	Path to the resourceFile, which holds the UUIDs of the resources being assigned to the distributor (more on the resourceFile below)
distributionType	Type of distribution that is being assigned for the resources for this distributor. The only two values allowed are buy and loan???.
available	OPTIONAL. Contains the number of simultaneous loans that this distributor has the rights to loan out. If returnable = false, this element has no meaning. The content of the available element must be an int.
returnable	OPTIONAL Specifies whether the book in the request is returnable. This value should be false for buys and true for loans. The content of the returnable element must be true or false (currently, all lower-case as the API is case-sensitive).
userType	OPTIONAL user|passhash
permissions	OPTIONAL Element that contains the permissions grammar for this book.

Here is a sample XML Configuration file:

```
<?xml version="1.0" encoding="UTF-8"?>
   <request xmlns="http://ns.adobe.com/adept">
   <serverURL>http://your.server.com/admin/ManageDistributionRights</serverUR
L>
   <distributionRights>
      <distributor>urn:uuid:88037e33-0e4d-4180-80ce-6c1d5ddf9cc9</distributor>
      <resourceFile>/Users/You/Documents/DistribTool/test-resource-
list.txt</resourceFile>
      <distributionType>loan</distributionType>
      <available>2</available>
      <returnable>false</returnable>
      <userType>user</userType>
      <permissions>
         <display />
         <play />
         <excerpt />
         <print />
      </permissions>
   </distributionRights>
</request>
```

The XML Configuration file must contain one or more complete distributionRights elements. The distributionRights elements may contain different or same distributor UUIDs, resourceFiles, and distributionTypes. This is useful if you want to use a single request to assign "buy" books and "loan" books to the same distributor, or if you want to assign one or more sets of books to multiple distributors. If a distributionRights element does not contain all of its children elements, it will be skipped (but other distributionRights elements that have all of the child elements present will be processed).

2.5.3 Resource File

The second external resource that is required for this tool to run is the resourceFile. The path to the resourceFile is specified in the resourceFile (relative path may be used, but remember that the path is relative to the tool's call directory, and not necessarily the XML Configuration file's directory). The structure of the resourceFile is simple. It is a text file that has UUIDs of resources that will be assigned to the distributor, separated by line breaks. There may only be one UUID per line. As only one resource UUID is allowed per request, this tool will send one request per UUID, meaning that if one request fails, the rest will continue to be processed.

Here is a sample text resourceFile rest-resource-list.txt:

```
urn:uuid:110faa58-c8eb-4be0-b05a-ca31908d499a
urn:uuid:dce65269-621e-40ec-8b70-9f73b4557b75
urn:uuid:7065168f-a6aa-4869-b4ef-58278b10f8d4
```

The UUIDs used in the file must correspond to books in the inventory on your admin server. Be sure you have permission to assign them to distributors.

This resource file can be generated from the QueryTool. See section on Using QueryTool.

2.5.4 Request generated by DistribTool to assign books

DistribTool internally generates a XML request based on the input parameters and the request is sent to the server for processing.

Table below describes the elements and their meanings.

Table 4. DistribTool XML Request Elements

Options	Purpose
request	Root element of the request. Includes the Adept namespace (xmlns="http://ns.adobe.com/adept") as the default namespace. Attribute “action” is set to "create", signifying that a new assignment is being made. Attribute “auth” is set to "builtin", signifying that the built-in distributor should be the one that assigns the resources to the distributor whose UUID is provided in the distributor element.
distributionRights	Empty element that contains child elements: distributor, resourceFile, and distributionType. The XML Configuration file must contain one or more distributionRights elements

distributor	the UUID of the distributor to whom the resources specified by resourceFile will be assigned.
resource	UUID of the resource that will be assigned to the distributor. Only one resource may be assigned per request.
distributionType	Type of distribution that is being assigned for the resources for this distributor. The only two values allowed are buy and loan???.
available	OPTIONAL. Contains the number of simultaneous loans that this distributor has the rights to loan out. If returnable = false, this element has no meaning. The content of the available element must be an int.
returnable	OPTIONAL Specifies whether the book in the request is returnable. This value should be false for buys and true for loans. The content of the returnable element must be true or false (currently, all lower-case as the API is case-sensitive).
userType	OPTIONAL user|passhash
permissions	OPTIONAL Element that contains the permissions grammar for this book.
nonce	Base64-encoded Nonce, which is a quasi-unique transaction identifier that is created from the system time at the time that this tool is called and an incremented counter.
Hmac	Base64-encoded HMAC. This is a security feature for signing the request. To learn more about the HMAC and how it is generated, see the class description for XMLUtil Class.
expiration	W3CDTF date and time of the expiration of this request. The expiration is set to be EXPIRATION_INTERVAL* minutes from the current system time.

* EXPIRATION_INTERVAL is set to 15 seconds in the DistribTool.java code.

A sample request created:

```
<?xml version="1.0" encoding="UTF-8"?>
  <request action="create" auth="builtin" xmlns="http://ns.adobe.com/adept">
   <distributionRights>
      <distributor>urn:uuid:bfd6065e-05e7-4d4b-a5fc-ae171cca44f0</distributor>
      <resource>urn:uuid:d07cd8da-ab8c-491d-87da-b63735712dd3</resource>
      <distributionType>loan</distributionType>
      <available>2</available>
      <returnable>false</returnable>
      <userType>user</userType>
      <permissions>
         <display />
         <play />
         <excerpt />
         <print />
      </permissions>
   </distributionRights>
   <nonce>UGKx2ANBr7ytv1Dq/9ZzbQ==</nonce>
   <expiration>2012-03-28T13:41:16-05:00</expiration>
   <hmac>9RYj1o2xMHH6oGTD6JKzs3UBgGU=</hmac>
  </request>
```

2.5.5 Errors

CANNOT ADD OR UPDATE A CHILD ROW: A FOREIGN KEY CONSTRAINT FAILS

```
There was an error with the Request
<error xmlns="http://ns.adobe.com/adept" data="E_ADEPT_DATABASE http://cyue-
win7.dlogics.com:8081/admin/ManageDistributionRights
Cannot%20add%20or%20update%20a%20child%20row:%20a%20foreign%20key%20constrain
t%20fails%20(`cyuetest`.`assignedresource`,%20CONSTRAINT%20`assignedresource_
ibfk_2`%20FOREIGN%20KEY%20(`resourceid`)%20REFERENCES%20`resourcekey`%20(`res
ourceid`))"/>
```

If this error occurs, it usually means that this book is already assigned to the distributor. The book has to be removed first from the distributor before it can be added again.

3 Developing Your Own Tools

ACS software comes with all the tools necessary to set up a book inventory, but, as you can see, it is not suitable for an automated or distributed workflow out of the box. For many ACS operators, you may want to implement your own packaging tool that include the functionalities from the Upload Test tool and tasks you need to perform with Admin Console GUI.

ACS software provides source code and helper classes to help you with the development of such workflow requirements.

This section will point out the key elements of this development work to help answer questions commonly asked by our customers.

3.1 What Can You Do?

Generally speaking, you can write a GUI front end to accept parameters for packaging books, keeping it either strictly internal or making it outward facing to allow publishers or other groups to enter book information and upload the actual book in its original form. The GUI part can be web based or a desktop client.

This GUI can replace the Upload Test tool command in the packaging workflow.

After the book is packaged, it may not be ideal to use Admin Console to manually assign books to distributors and give them additional permissions. You can, as part of your automated workflow, also implement appropriate calls to programmatically perform these functions.

3.2 What do you need?

Reference: ACS Technical Reference Manual Section1.2 "Packaging" and User Manual Section 11 "Packaging Books with Content Server"

Almost all of these tasks are done via simple web services which make your implementation easy and platform independent. The web services use simple HTTP requests and responses.

One of the best resources to learn how to create these requests is the API Inspector in the Admin Console. Any operation performed in the Admin Console, as well as the server's response, will be displayed in the Inspector at the bottom of the screen. Combining this with the ACS Technical Reference should give you a very good idea of various objects and calls you can use.

3.2.1 Creating HTTP Requests

Understanding how to perform operations to interact with ACS begins with understanding the HTTP requests it uses. The User Manual has details on the specifics of formatting, and the Technical Reference goes into far more detail than this document will.

For a more complete XML request example, see Section 5. In its most basic form, the packaging request has the following form. In any case, it should be sent to `http://[YOUR WEBSITE]/packaging/Package`:

```
<package xmlns="http://ns.adobe.com/adpet”>
  <data>
    [BOOK DATA]
  </data>
  <expiration>
    [REQUEST EXPIRATION]
  </expiration>
  <nonce>
    [UNIQUE IDENTIFIER]
  </nonce>
  <hmac>
    [HMAC SIGNATURE]
  </hmac>
</package>
```

The data element is the entire EPUB or PDF file in one base64-encoded string. The expiration element is the time at which the request expires in W3CDTF (YYYY-MM-DDTHH:MM:S-TZ:TZ) format. This should use the current time of the system doing the packaging, plus some small value on the order of a few minutes. The nonce value should be, for security purposes, a unique value for every request sent and should also be base64-encoded. The HMAC is the SHA1 hash value of the builtin distributor's shared secret.

3.2.2 Signing the Request

As was stated above, each HTTP request needs to have an “HMAC” element. The best way to learn how this is done is to view the source code in snippets-2011_09_23.zip. The file XmlUtils.cs in “csharp/signers/AdeptUtils” contains the code for signing HTTP requests. For PHP source code, see the example provided by Datalogics. For Java code, see the source used to make UploadTest and other tools mentioned earlier in this document.

To sign a request using the provided code, the request string and built-in distributor’s shared secret must be provided, with the shared secret in a base64-decoded byte array. Once that information is passed to the provided functions, the request will be loaded into an XML object, with an HMAC element appended as the HMACSHA1 hash of the serialized request.

Most requests should be signed with the built-in distributor’s shared secret, but some may need the specific distributor’s shared secret. Make sure that you use the right one.

3.2.3 Final Steps

Once the request has been sent and a good response is received from the server, the book has been packaged. The book will be encrypted with Adobe DRM and it will be added to the database along with its permissions, metadata, and thumbnail. Also, a copy of the encrypted book and its thumbnail will be copied to the media server location specified in your packaging configuration file.

After the book has been packaged, however, there are still a few steps that need to be taken before it's ready for your e-commerce site. Primarily, a newly packaged book needs to be added to a distributor. After that, a book is usually assigned distributor rights. These steps, as with packaging and many post-packaging operations, can be done through HTTP requests. The best place to learn how to create these requests is the API Inspector in the Admin Console. Any operation performed, as well as the server’s response, will be displayed in the Inspector at the bottom of the screen.

Once the above tasks are completed, the changes to the distributor’s catalog need to be reflected on your website. You can query the database directly for this information. The “resource” and “resourceitem” tables have all the information you’ll need for each book. No sample code for this method is provided, since the sample store uses the catalog file from the Admin Console.

3.3 Source Code Samples and Helper Classes

The Java source code for the Upload Test tool can be found in UploadTest-1_2.zip in the ACS distribution.

XML Signing helper source code for Java for the signing requests can be found in the same source as the Upload Test source. Helper code for C# can be found in snippets-2011_09_23.zip. Helper code for PHP can be found in snippets-2011_09_23.zip as well, but we provide an updated XMLSigningSerializer.php file, which fixes some issues with the old code, and a small test PHP file to test the request. These are distributed with the ACS software. For evaluation customers, please email Datalogics for the updated version if needed.

4 Replacing a Book

When packaging a book, either with the Upload Test tool or with your own HTTP requests, the default action for the packaging server to take is to *add* the book to the database. Even if there is an existing book with the same title already packaged, the book will still be added alongside it as a new book.

If you want to replace an existing book with a new version, you will have to tell the packaging server that it should replace the book and also what the resource ID of the book to be replaced is. To do that, two elements need to be added to the packaging HTTP request. If you're using the Upload Test tool, you must pass the "`-xml`" argument to the tool and add the lines to the corresponding XML file.

```
<package xmlns="http://ns.adobe.com/adpet”>
  <action>[ACTION]</action>
  <resource>[RESOURCE ID]</resource>
  ...
</package>
```

The value for `[ACTION]` should be either "add" or "replace". If the action element is omitted, "add" is used by default. The value for `[RESOURCE ID]` should be the ID of the book to be replaced. When the "add" option is used and this element is omitted, ACS assigns the ID automatically in UUID URN format (urn:uuid:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx). If you are replacing a book, this is the value that should be used, including the "urn:uuid:" prefix.

To replace a book, you will also need to set an option in the packaging configuration file.

```
com.adobe.adept.packaging.allowURLCollisionsOnReplace=true
```

You can place this line anywhere in the file. This will tell the packaging server that it should overwrite books of the same name instead of throwing an error. After making the change, be sure to restart Tomcat so the change can take effect.

5 Troubleshooting/FAQ

What do I do if I run out of memory when running the Upload Test tool?

In some cases, specifying a larger heap size for the Upload Test tool can help. For the `-Xmx` argument, use a large number such as 1024 or greater.

If you're trying to package a particularly large book, you may not want to include the book's actual data in the HTTP request. In that case, you can use a datapath element instead of the regular data element. The datapath element must have the full path to the book to be packaged and the book must reside on the same system as the packaging server itself.

For more information, see section 3.5, "Working with Very Large Files", in the ACS Quick Start Guide.

Why am I getting the error "E_ADEPT_DISTRIBUTOR_AUTH"?

When using the Upload Test tool, this error occurs when the password provided is missing or incorrect.

When creating your own HTTP request from your own tool, you need to make sure you are using the builtin distributor's shared secret and are creating the HMAC correctly. This shared secret is based off of your Admin Console password. When the password is changed, so too will the shared secret. You can find the shared secret's value in the "distributor" table of your database.

Why am I getting the error "E_ADEPT_INTERNAL [request URL] null"?

While there may be multiple causes of this error, the most common cause is having an incorrectly defined namespace. Notice that while the Adobe namespace (first defined in the <package> element) does not have a trailing forward slash ('/'), the DC namespace (typically first defined in the <metadata> element) *does* have this trailing forward slash. It is not optional for the DC namespace to have this character.

Why doesn’t my packaged book appear with a title in the Admin Console?

Some books will have their metadata built in and can be read by the packaging server. For those that don’t, metadata needs to be provided. When using the Upload Tool, include an XML file with the -xml argument or provide the metadata fields individually. When creating your own HTTP request, include the appropriate metadata elements in the request.

How can I view my packaged book?

Once you have a book packaged, encrypted, and moved to the media location, you can no longer view the book in standard PDF or EPUB viewers or by simply double-clicking it. The book must be fulfilled and opened in a reader compatible with Adobe DRM. The sample store included with ACS is capable of fulfilling books that you have packaged. Once the book is downloaded from the store, it can be opened in Adobe Digital Editions. Upon starting ADE for the first time, you will be prompted to enter your Adobe ID. An Adobe ID can be obtained from adobe.com. Once your ID is entered into ADE, the book can be opened and read normally.

What is the complete form of the XML packaging request?

Each element should be fairly self-explanatory, but is defined in the User Manual. Note that the namespace string used in the metadata tag needs to have the trailing “/” at the end.

```
<package xmlns="http://ns.adobe.com/adept/">
  <action>add|replace</action>
  <resource>resource ID</resource>
  <voucher>voucher ID for GBLink fulfillment</voucher>
  <resourceItem>resource item index</resourceItem>
  <fileName>file name to use for this packaged resource</fileName>
  <location>upload location for packaged resource (file name or FTP URL)
</location>
  <src> download location for the packaged resource (HTTP URL)</src>
  <metadata xmlns:dc="http://purl.org/dc/elements/1.1/">
    <dc:title>Book Title</dc:title>
    <dc:description>Book Description</dc:description>
    <dc:language>Book Language</dc:language>
    <dc:creator>Book Creator</dc:creator>
    <dc:publisher>Book Publisher</dc:creator>
    <dc:format>Book mimetype</dc:format>
    <dc:identifier>Book identifier</dc:identifier>
  </metadata>
  <permissions>
    <display>rights elements</display>
    <play>rights elements</play>
    <excerpt>rights elements</excerpt>
    <print>rights elements</print>
  </permissions>
  <dataPath>instead of data element, specifies local eBook location on
packaging server</dataPath>
  <data>Base64-encoded book bytes </data>
  <thumbnailLocation>thumbnail upload location (file name or FTP
URL)</thumbnailLocation>
  <thumbnailData>Base64-encoded thumbnail bytes </thumbnailData>
  <expiration>W3CDTF expiration </expiration>
  <nonce>Base64-encoded nonce </nonce>
  <hmac>Base64-encoded HMAC </hmac>
</package>
```

Where can I go for more help and information?

For more information on any of these points, you can refer to the User Manual or Technical Reference included with ACS. Source code and code snippets are available in the ACS package as well.

You can also check our knowledgebase http://kb.datalogics.com/knowledgeHome for explanations of errors and other general information. If there is something you can't find, let us know. We will make an effort to add articles to better answer your questions. Contact us at acs_support@datalogics.com for any questions that you still need answered.