
Parallels

Application Packaging Standard 1.1 Catalog

Developer's Guide and API Reference

Revision 1.0.01



(c) 1999-2009

Contents

Introduction	4
About APS Catalog	5
About This Guide	5
Useful Links	6
Typographical Conventions	6
Feedback	7
APS Catalog API Overview	8
Technical Details	8
APS Catalog Resources	9
APS Catalog API Operations	12
Browsing APS Catalog	13
Retrieving Repository Index	13
Discovering Repository Feed	14
Retrieving Repository Feed and Displaying Its Content	14
Advanced Browsing	14
Filters	15
Search Descriptions	17
Service Operations	17
Adding Content to APS Catalog	21
Adding Package to Catalog Directly	21
Adding Package via Packager Administration Panel	22
APS Catalog API Reference	23
Indexes	24
Index URLs	24
Service Index	25
Repository Index	26
Vendor Index	29
Application Index	31
Application Version Index	33
Package Index	34
Package Resources Index	35
Feeds	36
Feed URLs	36
Sample	37
Elements Reference	38
Search Descriptions	41
Search Description URLs	41
Search Description Arguments	42
Operations on Content Resources	45
ADD Package	45
RETRIEVE Operations	47

CHAPTER 1

Introduction

This chapter provides general information about Application Packaging Standard Catalog and this guide.

In This Chapter

About APS Catalog	5
About This Guide	5
Useful Links.....	6
Typographical Conventions.....	6
Feedback	7

About APS Catalog

APS Catalog is a repository of APS packages with search capabilities. A client application can be implemented to automate operations performed over APS Catalog. Available operations are following: discovering applications and updates, and adding new packages.

About This Guide

The purpose of this document is to provide instructions on how to use the APS Catalog API. The document is intended for developers whose aim is to write client applications for APS Catalog. Readers of this document should have a basic understanding of the Atom syndication format and OpenSearch 1.1, or at least they should be familiar with XML basics.

Important! This document covers only APS Catalog 1.1 API. If you use other version of the service, refer to the corresponding document.

Abbreviations, Definitions and Conventions

- *Application Packaging Standard* or *APS*. Application packaging format for site applications. This format enables to distribute and market site applications as a SaaS solution.
- *APS Package*. A package that conforms to the APS.
- *Client*. APS Catalog client application.
- *Feed*. An XML document that conforms to the Atom syndication format.
- *Feed page* is a feed that is a part of a paged feed. Terms *paged feed* and *feed* have the same meaning.
- *Enabled package*. A package that is validated by APS Catalog team or authorized packager. Only enabled packages are accessible through APS catalog interface.
- *XPath* language is used to describe the structure of XML documents and to refer to the XML elements and attributes in the text.
- [RFCNNNN] construction is used to refer to an RFC document. Here *NNNN* stands for the RFC number.

Useful Links

- *HTTP 1.1 Protocol Specification, [RFC2616]*
<http://www.ietf.org/rfc/rfc2616.txt>
- *Atom Syndication Format*
<http://www.ietf.org/rfc/rfc4287.txt>
- *Feed Paging and Archiving*
<http://www.ietf.org/rfc/rfc5005.txt>
- *OpenSearch 1.1*
<http://www.opensearch.org/Specifications/OpenSearch/1.1>
- *Application Packaging Standard*
<http://www.apsstandard.com>
- *Site Application Certification Criteria*
<http://apsstandard.com/r/doc/certification-criteria/index.html>
- *XPATH Specification*
<http://www.w3.org/TR/xpath>
- *XSL Transformations*
<http://www.w3.org/TR/xslt>
- *APS Application Categories*
<http://www.apsstandard.com/r/doc/aps--application-categories.pdf>

Typographical Conventions

Before you start using this guide, it is important to understand the documentation conventions used in it.

The following kinds of formatting in the text identify special information.

Formatting convention	Type of Information	Example
Special Bold	Items you must select, such as menu options, command buttons, or items in a list.	Go to the System tab.
	Titles of chapters, sections, and subsections.	Read the Basic Administration chapter.
<i>Italics</i>	Used to emphasize the importance of a point, to introduce a term or to designate a command line placeholder, which is to be replaced with a real name or value.	The system supports the so called <i>wildcard character</i> search.

Monospace	The names of files, and directories; XPATH expressions.	The license file is located in the /docs/common/licenses directory.
Preformatted	On-screen computer output in your command-line sessions; source code in XML, C++, or other programming languages.	<pre># ls -al /files total 14470</pre>
Preformatted Bold	XHTML elements described in a specific elements reference	<pre></pre>

Feedback

If you have found a mistake in this guide, or if you have suggestions or ideas on how to improve this guide, please send your feedback using the online form at <http://www.parallels.com/en/support/usersdoc/>. Please include in your report the guide's title, chapter and section titles, and the fragment of text in which you have found an error.

CHAPTER 2

APS Catalog API Overview

This chapter provides description APS Catalog API features. It is divided into the following sections:

- Technical Details (on page 8). Contains information on protocols and formats supported by APS Catalog API.
- APS Catalog Resources (on page 9). Describes resources provided by APS Catalog through API.
- APS Catalog API Operations (on page 12). Lists operations on APS Catalog resources.

In This Chapter

Technical Details	8
APS Catalog Resources.....	9
APS Catalog API Operations.....	12

Technical Details

A client interacts with APS Catalog by issuing HTTP GET/POST requests, and retrieving HTTP packets with status codes and APS Catalog resources. A resource can be a binary file (for example, a package) or an XML-based message. Each XML message conforms to one of the following formats:

- Atom feed [RFC 4287] with the following extensions:
 - OpenSearch 1.1
 - Feed paging [RFC 5005]
 - Native APS Catalog extensions describing APS package metadata
- XHTML format that describes APS catalog structure.
- Native XML-based format that describes APS catalog structure. For details, refer to the APS Catalog Resources section (on page 9) further in this guide.

APS Catalog Resources

This section describes resources that a client accesses through APS Catalog API. The resources are divided into the following types:

- Index. Contains information about APS Catalog hierarchy structure.
- Feed. Contains information about the APS Catalog content.
- Search description. Contains instructions on how to filter the APS Catalog content.
- Package bits. Represent a package file.
- Package metadata. Hold package metadata.
- Package resources. Comprise additional files included into a package.
- Package certificate. Holds a package certificate.

Note: APS Catalog represents each package as three resources: package bits, package metadata, and package resources.

The following diagram depicts the organization of APS Catalog resources:

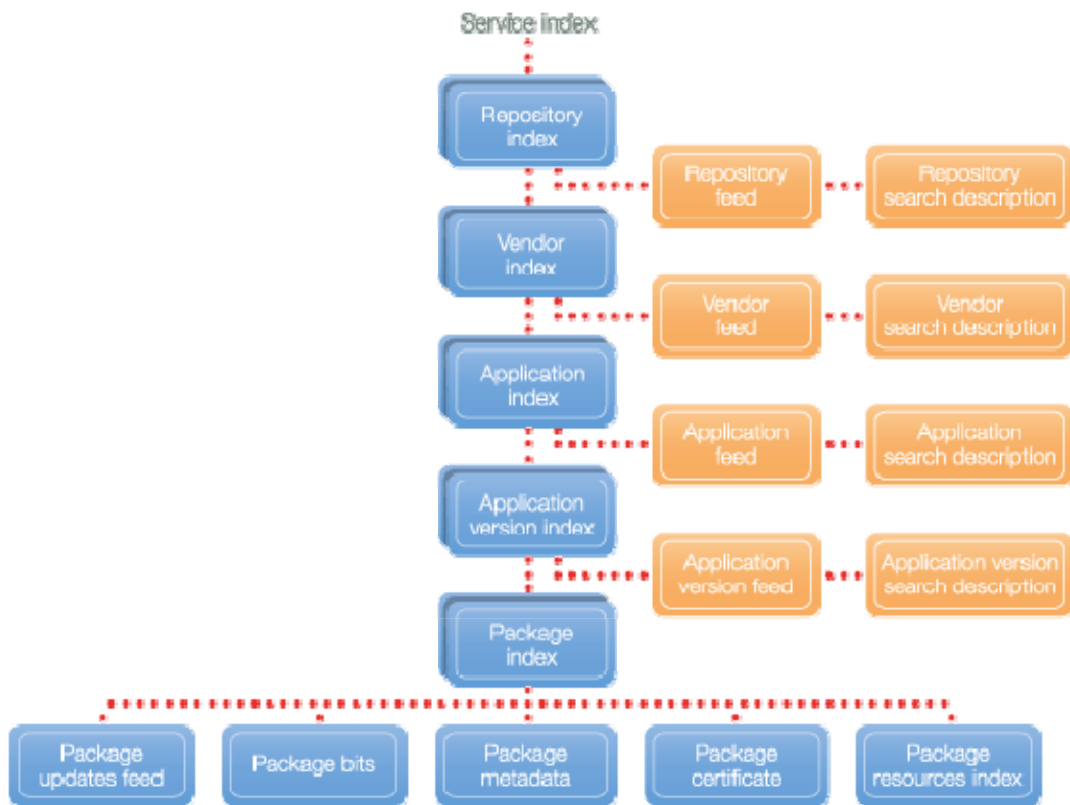


Figure 1: APS Catalog resources

The diagram shows that all resources are directly or indirectly linked with *indexes* - read-only XHTML data that hold information about APS Catalog hierarchy structure.

The following table shows references that are provided by a specific index.

Index Name	Link to Parent index	Link to Child indexes	Feed link	Link to package bits	Link to Package metadata	Link to package certificate	Link to package resources
Service index		+	+				
Repository index	+	+	+				
Vendor index	+	+	+				
Application index	+	+	+				
Application Version index	+	+	+				
Package index	+	+	+	+	+	+	
Package Resources index	+						+

In terms of file systems, indexes describe APS Catalog directories, each containing links to its content, parent directory, and subdirectories. A client can request an index at a specific URL. For details on index URLs and elements included in indexes, refer to the Indexes section (on page 24) of the API Reference.

As noted, almost every index has a feed link. If indexes represent the structure of APS Catalog, feeds represent its content. More precisely, a feed is composed of entries, each providing details on a single package. The details include a package name, vendor, version, etc.

APS Catalog provides five read-only Atom feeds:

- *Repository* feed. Contains details on each package within a repository.
- *Vendor* feed. Contains details on each package produced by a specific vendor.
- *Application* feed. Contains details on each package of a specific application. More precisely, each entry in this feed represents a package that differs from other packages by a packager, application version, or both.
- *Application Version* feed. Contains details on each package of the same application version issued by different packagers.
- *Package Updates* feed. Contains details on each package that updates a specific application. If a package does not have any updates, the Package Updates feed is empty.

A client can request a feed at the URL retrieved from a corresponding index. There are two features of how APS Catalog operates with feed requests.

- APS Catalog returns paged feeds. This means that APS Catalog divides a feed into parts before sending it to a client. This enables such client to request only a part of data, display it, and request the next part only if a user requests it. More precisely, a paged feed is a set of linked feeds that together contain all entries requested by a client. We will refer to each such paged feed part as to a *feed page*. Each feed page has links to the first feed page and the last feed page. Additionally, each feed page contains a link to the previous feed page and the next feed page (if these pages exist). Hereafter in this guide, terms "paged feed" and "feed" have the same meaning.



Figure 2: Structure of a feed

For details on paged feeds, refer to [RFC 5005].

- APS Catalog returns only modified feeds. This means that APS Catalog can return only feed entries that were modified since the last client's request. To fetch only these entries, a client must specify the last time of feed request in the *If-Modified-Since* parameter nested in the request header. If a client does not provide this information, APS Catalog returns all feed entries. For details on the *If-Modified-Since* parameter, refer to the 14.25 section of [RFC 2616]. Note, that to display all feed entries instead of only updated ones, a client must somehow cache all retrieved entries.

For details on feed URLs and XML structure, refer to the Feeds section (on page 36) of the API Reference.

A client can limit the number of entries in a requested feed by adding specific arguments to the feed request. For instance, a client can request a Repository feed restricted to entries which describe only Gold certified packages (for details on certification levels, refer to the Site Application Certification Criteria document). These arguments are feed specific. APS Catalog stores the request arguments in resources called *search description documents*. Hereafter in this guide, we will refer to these resources as to *search descriptions*. One search description corresponds to a single feed. Search descriptions conform to the search description format. For details on search descriptions format, see <http://www.opensearch.org/Specifications/OpenSearch/1.1>.

APS Catalog provides the following search descriptions:

- *Repository* search description
- *Vendor* search description
- *Application* search description
- *Application Version* search description

As it is seen from the list, only four feeds have search descriptions, namely: Repository, Vendor, Application and Application Version.

A client can request a search description at a specific URL. For details on search description URLs and XML structure, refer to the Search Descriptions section (on page 41) of the API Reference.

APS Catalog API Operations

To use the API, a client sends HTTP requests to APS Catalog that instructs it to perform a *RETRIEVE* or *ADD* operation over a resource.

- A client performs *RETRIEVE* operations to obtain a specific index, feed, package source, package metadata, package certificate, or search description. To retrieve a resource, a client issues an HTTP GET request to the resource URL.

Important: APS Catalog processes feed requests differently from other requests. For details, refer to the APS Catalog Resources section (on page 9) earlier in this guide.

- A client performs *ADD* operation to add a package to APS Catalog. Available methods are:
 - A client submits the *packageSubmitForm* form to APS Catalog.
 - A client submits and HTTP POST request to `/packager/upload/` URL.

For details on the Add operations, refer to the Adding Content to APS Catalog section (on page 21) further in this guide.

CHAPTER 3

Browsing APS Catalog

The simplest way to browse APS Catalog is to display all its contents. To perform this, a client must do the following:

- 1 Retrieve a Repository index.
- 2 Discover the Repository feed.
- 3 Retrieve the feed and display its content to end users.

This chapter provides in-depth instructions on how to perform each of these steps.

You can improve client flexibility in representation of APS Catalog resources making it filter the APS Catalog content. Such filtering enable end users to search for something particular in APS Catalog like "applications of a specific vendor". A client can narrow search results by providing additional search parameters. These parameters make user requests more precise, and reduce the traffic between the client and APS Catalog. The guidelines on how to implement these extensions are also provided in this chapter. For details, see the Advanced Browsing section (on page 14).

Some operations described in this chapter suppose parsing XML data for specific elements. To identify such elements, we will use XPATH expressions.

In This Chapter

Retrieving Repository Index	13
Discovering Repository Feed	14
Retrieving Repository Feed and Displaying Its Content.....	14
Advanced Browsing	14

Retrieving Repository Index

It is supposed that a client knows the Service index URL. For instance, this URL is <http://www.example.com/aps/>.

To retrieve a Repository index, the client must perform the following steps:

- 1 Request the System index by issuing the HTTP GET request to <http://www.example.com/aps/>.
- 2 Parse the index for the following expression: `/a[@class='repository']`. The `href` attribute of each such element contains URL of a Repository index. The element content contains the repository name.
- 3 Request the required Repository index by issuing the HTTP GET request to a corresponding URL.

Discovering Repository Feed

To retrieve a Repository feed URL, a client must parse the Repository index for the following expression: `/a[@id='feedLink']/@href/text()`.

Retrieving Repository Feed and Displaying Its Content

To display feed content to end users, a client must perform the following steps:

- 1 Retrieve the feed by issuing an HTTP GET request to the feed URL.
 - a If APS Catalog returns the status code 304, fetch the feed from cache.
 - b If APS Catalog returns a feed, save the feed to cache.
- 2 Parse the requested feed page for navigation bar links:
 - First feed page link: `/link[@rel='first']/@href/text()`
 - Previous feed page: `/link[@rel='prev']/@href/text()`
 - Next feed page link: `/link[@rel='next']/@href/text()`
 - Last feed page link: `/link[@rel='last']/@href/text()`
- 3 Extract the `entry` elements from the feed and transform them to the required format. For details the `entry` element, refer to the Elements Reference section (on page 38) in API Reference.

Note: Steps 2 and 3 are repeated on each feed page request. Step 3 can be easily performed by using XSL transformations.

- 4 Render a web page with the navigation bar and transformation results and display the page to end users.

A client can create a navigation bar that includes links to all feed pages. This requires utilizing search descriptions. For details on how to add the navigation bar, refer to the Search Descriptions section (on page 17) further in this guide.

Advanced Browsing

To provide end users with advanced browsing capabilities, a client can use one of the following APS Catalog features (or their combination):

- Filters
- Search descriptions

Both filters and search descriptions affect the information on APS Catalog resources which is displayed to end users. By using filters/search descriptions, a client can reduce the incoming traffic value, and users can obtain more precise data. The difference between filters and search descriptions is that filters fetch particular feeds from APS Catalog indexes when forming a results set, while search descriptions contain rules that are applied to a certain feed (as request arguments) and affect the content of this feed.

The Filters and Search descriptions sections contain step-by-step instructions on how to filter the APS Catalog content. Some of these steps require additional explanations which are provided in a separate section called Service Operations.

Filters

A filter is a named rule which uses indexes to return only packages that meet a specific filtering criterion. For instance, to filter packages by vendors, a client must perform the following steps:

- 1 Retrieve vendor names and links to Vendor indexes from a Repository index.
- 2 Display vendors list to a user.
- 3 Retrieve an index corresponding to the vendor (after the user chooses one of the vendors).
- 4 Fetch a feed link from the index.
- 5 Display the feed content to the user.

This example outlines the routine of using such filters. It is as follows:

- 1 A client retrieves descriptions of lower-level indexes and links to these indexes from a current index. For instructions on how to do it, see the Retrieving Information from Indexes section (on page 18).
- 2 The client displays these descriptions as filter names to a user.
- 3 The user chooses one of the filter names.
- 4 The client extracts a feed link from an index corresponding to the filter name. For instructions on how to do it, see the Discovering Feeds section (on page 18).
- 5 The client displays feed content to the user. As all feeds have the same syntax, this step is performed using the same instructions as described in the Retrieving Repository Feed and Displaying Its Content section (on page 14).

According to the APS, each package pertains to one or more packagers. A client can filter packages by their packagers. To do it, a client must perform the following steps:

- 1 Retrieve a Repository index.
- 2 Extract a list of packagers by parsing the index for the `//select[@id='packager']`.
- 3 Generate a feed link to each packager. Each such link looks as follows:
`<formAction>?<selectName>=<optValue>`
 Here
 - `<formAction>` stands for `//form/@action/text()`

- `<selectName>` stands for `//select/@name/text()`
- `<optValue>` stands for `//select[@id='packager']/optgroup/option/@value/text()`.

For example, we have the following part of a Repository index:

```
<label for="packager">Packager</label>
  <select id="packager" name="packager">
    <option>www.parallels.com</option>
  </select>
```

After parsing this part, a client must get a www.parallels.com packager with a feed link `../1.atom?packager=www.parallels.com`.

4 Display the feed content on a user's request.

Note: feed content can be displayed with a browser. In this case, keep in mind that different browsers may handle feeds differently.

Remarks

You can use the same mechanism to filter the APS Catalog content by other parameters like target architectures, target platforms, etc.

APS Catalog offers two additional URIs to get lists of vendors and categories in XML formats. You can use these lists to form feed links that filter the APS Catalog content by vendor or category.

URI	Description	Result example
/categories	Returns list of all categories. There is at least one accessible package in each category	<pre><categories> <category id="Back office" package_count="9" name="Back office"> <category id="Back office/Manufacturing Solutions" package_count="2" name="Manufacturing Solutiocount="1" name="Accounting & Financial" /> ... </categories></pre>
/vendors	Returns list of all vendors. Each vendor provides at least 1 accessible package.	<pre><vendors> <vendor id="aerialchat.sourceforge.net" /> <vendor id="anyinventory.sourceforge.net" /> ... </vendors></pre>

A client can perform even more complex filtering. To build this functionality into a client, you can use filters together with search descriptions, or request index resources with additional arguments. For details on search descriptions, refer to the next section. For details on what arguments can be added to index resource URL, refer to the Indexes section (on page 24).

Search Descriptions

Search description is an XML resource linked with a feed. This resource contains templates of feed requests that cause APS Catalog to tweak the content of a feed before returning it to a client. Each such template differs by arguments added to a feed URL. For instance, to get a feed page with 20 entries, a client must issue the following HTTP GET request: `<feed_URL>?pageSize=20`. To get the second feed page with 20 entries per page, a client must issue the following request: `<feed_URL>?pageSize=20&page=2`. These two arguments are common for each feed linked with a search description. For description of feed request arguments, refer to the Search Descriptions section (on page 41) further in this guide. A typical usage of arguments nested in search descriptions is illustrated in the next paragraph.

How to create an extended navigation bar

A client can use the *page* and *pageSize* arguments to create a navigation bar that comprises links to all feed pages. It is supposed that such client has already discovered a feed. To generate the bar, the client must do the following:

- 1 Define the number of entries to be displayed per feed page. Let *X* denotes this number.
- 2 Retrieve the first feed page by issuing HTTP GET request `<feed_URL>?pageSize=X`.
- 3 Retrieve the total number of entries in the feed by parsing the first feed page for `//feed/opensearch:totalResults/text()`. Let *Y* denote this number.
- 4 Calculate the number of feed pages by rounding Y / X .
- 5 Create links to each feed page. Each link will look like this: `<feed_URL>?pageSize=X&page=N`. Here *N* stands for a feed page number.

Remarks

A client can check if search description arguments are updated by parsing the corresponding search description.

Feed request arguments can be used in combination with filters to create custom filtering criteria. For example, a client can display Gold certified packages pertaining to a specific APS category. We recommend you to examine the XML structure of each search description to make use of the full range of functionality provided by them. For details on search descriptions XML structure, see the API Reference, section Search Descriptions section (on page 41).

Service Operations

This section describes how to perform operations related to filtering of APS Catalog content. These operations include discovering a specific feed and extracting specific data from indexes.

Discovering Feeds

Each APS Catalog index (except for Package Resources) has a feed link. To extract the feed link from an index, parse such index for `//a[@id='feedLink']`.

Package Updates feed differs from other feeds by its semantics. This feed contains a list of packages that update an application, rather than a list of application packages. To indicate this fact, a client must use `//a[@id='updates']` to retrieve the Package Updates feed link from a Package index.

Retrieving Information from Indexes

To create a filter, a client must fetch specific data from indexes (filter names, links to sub-indexes, etc). The data is nested in specific `<A>` tags. Each such tag provides information on an APS Catalog resource. The tag content is the human-readable resource name, the `href` attribute stores a link to the resource, and the ID/class attributes identify the tag. In most cases, the resource is a sub-index, and its name is used as the name of a filter, or the name of an APS Catalog hierarchy level. However, some indexes provide links to other resources like packages, package certificates, etc.

This section explains what expressions can be used to fetch these `<A>` tags from indexes. The section starts with parsing the System index to show that all APS Catalog resources can be recursively derived from this index.

- **Getting Data from the Service Index**

Information on Repository indexes can be extracted from the Service index.

Expression: `//a[@class='repository']`

For details on the Service index XML structure, refer to the Service Index section (on page 25) in the API Reference.

- **Getting Data from a Repository Index**

From a Repository index, a client can extract the following information:

- Info on a Vendor index
Expression: `//a[@class='vendor']`
- Info on categories
Expression: `//select[@id='category']/optgroup`
- Info on packagers
Expression: `//select[@id='packager']//option`
- List of available target architectures
Expression: `//select[@id='arch']//option`
- List of available target platforms
Expression: `//select[@id='platform']//option`
- List of available target OS
Expression: `//select[@id='os']//option`

- List of available package types
Expression: `//select[@id='type']//option`
- Info on package bits
Expression: `//a[@id='aps']`

For details on the Repository index XML structure, refer to the Repository Index section (on page 26) in the API Reference.

- **Getting Data from a Vendor Index**

Information on Application indexes can be extracted from a Vendor index.

Expression: `//a[@class='application']`

For details on the Vendor index XML structure, refer to the Vendor Index section (on page 29) in the API Reference.

- **Getting Data from an Application Index**

Information on Application Version indexes can be extracted from an Application index.

Expression: `//a[@class='version']`

For details on an Application index XML structure, refer to the Application Index section (on page 31) in the API Reference.

- **Getting Data from an Application Version Index**

Information on Package indexes can be extracted from an Application Version index.

Expression: `//a[@class='packager']`

For details on the Application version index XML structure, refer to the Application Version Index section (on page 33) in the API Reference.

- **Getting Data from a Package Index**

From a Package index, a client can extract the following information:

- Info on a Package Resources index
Expression: `//a[@id='resources']`

Note: If a package does not provide additional resources, the Package Resources index does not exist.

- Info on package metadata
Expression: `//a[@id='meta']`
- Info on package bits
Expression: `//a[@id='aps']`
- Info on a package certificate. Present only if the package is certified.
Expression: `//a[@id='certificate']`

For details on the Package index XML structure, refer to the Package Index section (on page 34).

- **Getting Data from a Package Resources Index**

From a Package resources index, a client can extract the following data:

- Info on package icons
Expression: `//a[@class='icon']`

- Info on package screenshots
Expression: `//a[@class='screenshot']`

For details on the Package resources index XML structure, refer to the Package Resources Index section (on page 35).

- **Getting a Parent Index Link**

To enable end users to return to upper level of APS Catalog hierarchy, a client can extract a parent index link. This link can be retrieved from all indexes except for the System index.

Expression: `//a[@id='parentLink']`

CHAPTER 4

Adding Content to APS Catalog

A client can allow end-users to add a package to APS Catalog with one of the following methods:

- Add a package to APS Catalog directly.
- Add a package via packager administration panel.

This chapter explains how to build this capability into a client.

In This Chapter

Adding Package to Catalog Directly.....	21
Adding Package via Packager Administration Panel	22

Adding Package to Catalog Directly

The sequence of actions performed by a client and a user to add a package to APS Catalog is as follows:

- 1 The user chooses an appropriate repository for the package, and passes the control to the client.
- 2 The client does the following:
 1. Retrieves the Repository index.
 2. Parses the index for a specific form
(`//form[@name='packageSubmitForm']`)
 3. Displays this form to end users. Form fields indicate information required for adding the package.
Before displaying this form, the client can transform it to meet the UI design requirements.
- 3 The user fills the form and submits it to APS Catalog.

Note: packages uploaded through APS Catalog form are *enabled* by default.

For details on form fields (HTTP POST arguments) and possible APS Catalog responses, refer to the ADD Package section (on page 45) in the API Reference.

Adding Package via Packager Administration Panel

Packager Administration Panel services are available to registered users only. To get login credentials to Administration Panel user should complete an application form with APS Catalog team. Then user manually or with his client forms an HTML POST request to `/packager/upload/` URL. Only *enabled* packages can be accessed through catalog interface. Along with login credentials user may get rights to enable uploaded packages. User could enable package from Packager Administration Panel or with upload request.

For details on POST arguments and possible APS Catalog responses, refer to the ADD Package section (on page 45).

CHAPTER 5

APS Catalog API Reference

This chapter contains information about indexes, feeds, and search descriptions - three types of resources that represent APS Catalog organization and contents. For details on a specific resource, refer to the appropriate section of this chapter. This chapter also provides descriptions of operations on APS Catalog content resources - the resources that are stored in APS Catalog.

In This Chapter

Indexes.....	24
Feeds.....	36
Search Descriptions.....	41
Operations on Content Resources.....	45

CHAPTER 6

Indexes

The subsections of this section describe the XML structure of APS Catalog indexes. Each description includes a sample of index and reference of index elements. The sample index elements that present in the reference are bolded.

In This Chapter

Index URLs	24
Service Index	25
Repository Index	26
Vendor Index.....	29
Application Index.....	31
Application Version Index.....	33
Package Index	34
Package Resources Index	35

Index URLs

Each index is available at one of the following URLs:

Index name	Index URL
Service	<ROOT_URL>/
Repository	<ROOT_URL>/ <APSver>/
Vendor	<ROOT_URL>/ <APSver>/ <VendorName>/
Application	<ROOT_URL>/ <APSver>/ <VendorName>/ <AppName>/
Application Version	<ROOT_URL>/ <APSver>/ <VendorName>/ <AppName>/ <AppVer>/
Package	<ROOT_URL>/ <APSver>/ <VendorName>/ <AppName>/ <AppVer>/ ? packager=<Packager>
Package Resources	<ROOT_URL>/ <APSver>/ <VendorName>/ <AppName>/ <AppVer>/resources

Note: If a package does not provide additional resources, the Package Resources index does not exist.

The placeholder strings mean the following:

- `<ROOT_URL>` - the APS Catalog URL;
- `<APSver>` - the APS version. Allowed values: 1, 1.1, all;
- `<VendorName>` - the name of a specific vendor;
- `<AppName>` - the name of a specific application;
- `<AppVer>` - the version of a specific application;
- `<Packager>` - the name of a packager.

URL Arguments

You can add the additional arguments to the index URL to narrow the results set. If the requested resources are not found, APS Catalog will respond with error code 404. For the list of available arguments return to the Search Description Arguments section (on page 42) further in this guide.

Service Index

Sample

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en"
dir="ltr">
  <head> <title>APS repositories</title> </head>
  <body>
    <p> <a href="apsRepo.atom" id="feedLink">Repositories feed</a>
  </p>
  <p class="repositoryList">
    <a href="1/" class="repository">APS version 1</a>
    <a href="1.1/" class="repository">APS version 1.1</a>
    <a href="all/" class="repository">APS all versions</a>
  </p>
  </body>
</html>
```

Elements Reference

Element	Description
/html/body/p/a[@id='feedLink']	Service feed link
/html/body/p[@class='repositoryList']/a[@class='repository']	Link to Repository index

Repository Index

Sample

```

<?xml version="1.0" encoding="utf-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en"
dir="ltr">
  <head>
    <title>APS Catalog</title>
  </head>
  <body>
    <p><a href=".." id="parentLink">Service Index</a></p>
    <p><a href="../1.atom" id="feedLink">Repository feed</a></p>
    <p class="vendorList">
      <a href="http://www.parallels.com/"
class="vendor">www.parallels.com</a><br />
      <a href="http://www.proxy2.de/" class="vendor">www.proxy2.de</a><br />
      <a href="http://www.phpwcms.de/"
class="vendor">www.phpwcms.de</a><br />
    </p>
    <form id="filterForm" action="../1.atom" method="get">
      <label for="name">Application name:</label>
      <input type="text" id="name" name="name" value="" /><br />
      <label for="category">Category</label>
      <select id="category" name="category">
        <optgroup label="Web">
          <option value="Web/Wiki">Wiki</option>
        </optgroup>
      </select>
      <br />
      <label for="packager">Packager</label>
      <select id="packager" name="packager">
        <option>www.parallels.com</option>
      </select>
      <br />
      <label for="cert">Certification level</label>
      <select id="cert" name="cert" multiple="multiple">
        <option value="none">Not certified</option>
        <option value="gold">Gold</option>
        <option value="silver">Silver</option>
      </select>
      <br />
      <label for="type">Package type</label>
      <select id="type" name="type">
        <option value="">Any</option>
        <option value="APS">APS</option>
        <option value="PVT">Parallels Container Templates</option>
      </select>
      <br />
      <label for="arch">Architecture</label>
      <select id="arch" name="arch">
        <option value="">Any</option>
        <option value="x86">x86</option>
        <option value="x86_64">x86_64</option>
      </select>

```

```

        <br/>
        <label for="platform">Platform</label>
        <select id="platform" name="platform">
            <option value="">Any</option>
            <option value="linux">Linux</option>
            <option value="windows">Windows</option>
        </select>
        <br/>
        <label for="os">Operating system</label>
        <select id="os" name="os">
            <option value="">Any</option>
            <option value="suse-10.2">SuSE 10.2</option>
            <option value="fedora-core-6">Fedora Core 6</option>
            <option value="Windows 2003 Server">Windows 2003
server</option>
        </select>
        <input type="submit" value="Filter" />
    </form>
    <form id="packageSubmitForm" action="." method="post"
enctype="multipart/form-data">
        <label for="name">Application package:</label>
        <input type="file" id="package" name="package" /><br />
        <input type="submit" value="Submit package" />
    </form>
</body>
</html>

```

Elements Reference

Element	Description
/html/body/p/a[@id='parentLink']	Link to System index
/html/body/p/a[@id='feedLink']	Repository feed link
/html/body/p[@class='vendorList']/a	Link to Vendor index
/html/body/form[@id='filterForm']	If this form is submitted with specific arguments (<i>category</i> , <i>packager</i> , or <i>cert</i>), APS Catalog returns one of the following: <ul style="list-style-type: none"> ▪ Feed with details on each package of a specific packager ▪ Feed with details on each package pertaining to a specific APS category ▪ Feed with details on each package of a specific certification level
/html/body/form[@id='filterForm']/label/select[@id='category']	Element that nests APS categories
/html/body/form[@id='filterForm']/label/select[@id='category']/optgroup/option/@value/text()	Category name
/html/body/form[@id='filterForm']/label/select[@id='packager']/option/text()	Packager name

<pre>/html/body/form[@id='filterForm']/label/select[@id='cert']/option/@value/text()</pre>	<p>Certification level</p>
<pre>/html/body/form[@id='filterForm']/label/select[@id='type']/option/@value/text()</pre>	<p>Package type</p>
<pre>/html/body/form[@id='filterForm']/label/select[@id='arch']/option/@value/text()</pre>	<p>Defines target architectures supported by a certain repository</p>
<pre>/html/body/form[@id='filterForm']/label/select[@id='platform']/option/@value/text()</pre>	<p>Defines target platforms supported by a certain repository</p>
<pre>/html/body/form[@id='filterForm']/label/select[@id='os']/option/@value/text()</pre>	<p>Defines target OS supported by a certain repository</p>
<pre>/html/body/form[@id='packageSubmitForm']</pre>	<p>Form for adding package to repository</p>

Vendor Index

Sample

```
<?xml version="1.0" encoding="utf-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en"
dir="ltr">
  <head>
    <title>www.proxy2.de - APS Version 1</title>
  </head>
  <body>
    <p><a href=".." class="repository" id="parentLink">APS version
1</a></p>
    <p><a href="http://www.proxy2.de/atom" id="feedLink">www.proxy2.de
package feed</a></p>
    <p class="applicationList">
      <a href="http://www.proxy2.de/advancedguestbook/"
class="application">advancedguestbook</a>
      <a href="http://www.proxy2.de/AdvancedPoll/"
class="application">AdvancedPoll</a>
    </p>
    <form id="filterForm" action="http://www.proxy2.de/atom" method="get">
      <label for="name">Application name</label>
      <input type="text" id="name" name="name" /><br />
      <label for="category">Category</label>
      <select id="category" name="category">
        <optgroup label="Web">
          <option value="Web/Wiki">Wiki</option>
        </optgroup>
      </select>
      <br />
      <label for="packager">Packager</label>
      <select id="packager" name="packager">
        <option value="http://www.parallels.com">www.parallels.com</option>
      </select>
      <br />
      <label for="cert">
        Certification level</label>
      <select id="cert" name="cert" multiple="multiple">
        <option value="none">Not certified</option>
        <option value="gold">Gold</option>
        <option value="silver">Silver</option>
      </select>
      <br />
      <input type="submit" value="Filter" />
    </form>
  </body>
</html>
```

Elements Reference

Element	Description
/html/body/p/a[@id='parentLink']	Link to Repository index
/html/body/p/a[@id='feedLink']	Vendor feed link

/html/body/p[@class='applicationList']/a	Link to Application index
/html/body/form[@id='filterForm']	<p>If this form is submitted with specific arguments (<i>category</i>, <i>packager</i>, or <i>cert</i>), APS Catalog returns the following data about packages of a specific vendor:</p> <ul style="list-style-type: none"> ▪ Feed with details on each package of a specific packager ▪ Feed with details on each package pertaining to a specific APS category ▪ Feed with details on each package of a specific certification level
/html/body/form[@id='filterForm']/label/select[@id='category']	Element that nests APS categories
/html/body/form[@id='filterForm']/label/select[@id='category']/optgroup/@label/text()	Category name
/html/body/form[@id='filterForm']/label/select[@id='packager']/option/text()	Packager name
/html/body/form[@id='filterForm']/label/select[@id='cert']/option/@value/text()	Certification level

Application Index

Sample

```
<?xml version="1.0" encoding="utf-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en"
dir="ltr">
  <head>
    <title>advancedguesbook - www.proxy2.de - APS Version 1</title>
  </head>
  <body>
    <p><a href=".." class="vendor"
id="parentLink">www.proxy2.de</a></p>
    <p><a href="../advancedguestbook.atom"
id="feedLink">advancedguestbook versions feed</a></p>
    <p class="versionList">
      <a href="2.4.3-3/" class="version">2.4.3-3</a>
    </p>
    <form id="filterForm" action="../advancedguestbook.atom"
method="get">
      <label for="category">Category</label>
      <select id="category" name="category">
        <optgroup label="Web">
          <option value="Web/Wiki">Wiki</option>
        </optgroup>
      </select>
      <br />
      <label for="packager">Packager</label>
      <select id="packager" name="packager">
        <option>www.parallels.com</option>
      </select>
      <br />
      <label for="cert">
        Certification level</label>
      <select id="cert" name="cert" multiple="multiple">
        <option value="none">Not certified</option>
        <option value="gold">Gold</option>
        <option value="silver">Silver</option>
      </select>
      <br />
      <input type="submit" value="Filter" />
    </form>
  </body>
</html>
```

Elements Reference

Element	Description
/html/body/p/a[@id='parentLink']	Link to Vendor index
/html/body/p/a[@id='feedLink']	Application feed link
/html/body/p[@class='versionList']/a	Link to Application Version index

<pre>/html/body/form[@id='filterForm']</pre>	<p>If this form is submitted with specific arguments (<i>category</i>, <i>packager</i>, or <i>cert</i>), APS Catalog returns the following data about packages of a specific application version:</p> <ul style="list-style-type: none"> ▪ Feed with details on each package of a specific packager ▪ Feed with details on each package pertaining to a specific APS category ▪ Feed with details on each package of a specific certification level
<pre>/html/body/form[@id='filterForm']/label/select[@id='category']</pre>	<p>Element that nests APS categories</p>
<pre>/html/body/form[@id='filterForm']/label/select[@id='category']/optgroup/@label/text()</pre>	<p>Category name</p>
<pre>/html/body/form[@id='filterForm']/label/select[@id='packager']/option/text()</pre>	<p>Packager name</p>
<pre>/html/body/form[@id='filterForm']/label/select[@id='cert']/option/@value/text()</pre>	<p>Certification level</p>

Application Version Index

Sample

```
<?xml version="1.0" encoding="utf-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en"
dir="ltr">
  <head>
    <title>2.4.3-3 - advancedguesbook - www.proxy2.de - APS Version
1</title>
  </head>
  <body>
    <p><a href=".." class="application"
id="parentLink">advancedguesbook</a></p><p>
    <a href="../2.4.3-3.atom" id="feedLink">advancedguestbook
2.4.3-3 packagers</a></p>
    <p class="packagerList">
      <a href="?packager=www.parallels.com"
class="packager">www.parallels.com</a>
      <a href="?packager=www.swsoft.com"
class="packager">www.swsoft.com</a>
    </p>
    <ul class="packagerList">
      <li><a href="?packager=www.swsoft.com&amp;type=aps"
class="packager">
        <abbr class="type" title="APS">APS package</abbr>
        <abbr class="packager"
title="www.parallels.com">Parallels, Inc</abbr>
        </a>
      </li>
      <li><a
href="?packager=www.swsoft.com&amp;type=pvt&amp;platform=linux&amp;arc
h=x86&amp;os=suse-10.2" class="packager">
        <abbr class="type" title="PVT">PVT package</abbr>
        <abbr class="platform" title="linux">Linux</abbr>
        <abbr class="os" title="suse-10.2">SuSE 10.2</abbr>
        <abbr class="arch" title="x86">Intel 386 or
compatible</abbr>
        <abbr class="packager"
title="www.parallels.com">Parallels, Inc</abbr>
        </a>
      </li>
    </ul>
  </body>
</html>
```

Elements Reference

Element	Description
/html/body/p/a[@id='parentLink']	Link to Application Version index
/html/body/p/a[@id='feedLink']	Application Version feed link
/html/body/p[@class='packagerList']//a	Links to Package indexes

/html/body/p[@class='packagerList']/a/abbr[@class='type']/text()	Package type
/html/body/p[@class='packagerList']/a/abbr[@class='packager']/text()	Packager name
/html/body/p[@class='packagerList']/a/abbr[@class='platform']/text()	Target platform
/html/body/p[@class='packagerList']/a/abbr[@class='os']/text()	Target OS
/html/body/p[@class='packagerList']/a/abbr[@class='arch']/text()	Target architecture

Package Index

Sample

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en"
dir="ltr">
<head>
<title>www.parallels.com - 2.4.3-3 - advancedguesbook -
www.proxy2.de - APS Version 1</title>
</head>
<body>

<p><a href=".." class="version" id="parentLink">2.4.3-3</a></p>
<a id="aps" href=".. /2.4.3-
3.aps?packager=www.parallels.com">Application package</a>
<a id="meta" href=".. /2.4.3-
3.meta?packager=www.parallels.com">APP-META</a>
<a id="certificate" href=".. /2.4.3-
3.certificate?packager=www.parallels.com">Application certificate</a>
<a id="updates"
href="updates.atom?packager=www.parallels.com">Package updates</a>
<a id="resources"
href="resources/?packager=www.parallels.com">Package resources</a>
<form id="certificateSubmitForm" action="." method="post"
enctype="multipart/form-data">
<label for="certificate">Application certificate:</label>
<input type="file" id="certificate" name="certificate" /><br
/>
<input type="hidden" id="packageID" name="packageID"
value="42" /><br />
<input type="submit" value="Submit certificate" />
</form>
</body>
</html>
```

Elements Reference

Element	Description
/html/body/p/a[@id='parentLink']	Link to Application Version index

/html/body/a[@id='aps']	Link to package bits
/html/body/a[@id='meta']	Link to package metadata
/html/body/a[@id='certificate']	Link to package certificate
/html/body/a[@id='updatesFeed']	Link to Updates feed
/html/body/a[@id='resources']	Link to Package Resource index
/html/body/form[@id='certificateSubmitForm']	Form for adding package certificate
/html/body/form[@id='certificateSubmitForm']/input[@id='packageID']/@value/text()	Package ID

Package Resources Index

Sample

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en"
dir="ltr">
<head>
<title>www.parallels.com - 2.4.3-3 - advancedguesbook -
www.proxy2.de - APS Version 1</title>
</head>
<body>
<p><a href="..?packager=www.parallels.com" class="packager"
id="parentLink">2.4.3-3</a></p>
<a class="screenshot" href="screenshot1.png">Admin panel</a>
<a class="screenshot" href="screenshot2.png">Guest view</a>
<a class="icon" href="icon.png">Application icon</a>
</body>
</html>
```

Elements Reference

Element	Description
/html/body/p/a[@id='parentLink']	Link to Package index
/html/body/a[@class='screenshot']	Screenshot of an application wrapped in the package
/html/body/a[@id='icon']	Icon of an application wrapped in the package

 CHAPTER 7

Feeds

The subsections of this section describe the XML structure of an APS Catalog feed.

In This Chapter

Feed URLs	36
Sample	37
Elements Reference	38

Feed URLs

Each feed is available at one of the following URLs:

Feed name	Feed URL
Service	<ROOT_URL>/apsRepo.atom
Repository	<ROOT_URL>/<APSver>.atom
Vendor	<ROOT_URL>/<APSver>/<VendorName>.atom
Application	<ROOT_URL>/<APSver>/<VendorName>/<AppName>.atom
Application Version	<ROOT_URL>/<APSver>/<VendorName>/<AppName>/<AppVer>.atom
Package updates	<ROOT_URL>/<APSver>/<VendorName>/<AppName>/<AppVer>/updates.atom

The placeholder strings mean the following:

- <ROOT_URL> - the APS Catalog URL;
- <APSver> - the APS version. Allowed values: 1, 1.1, all;
- <VendorName> - the name of a specific vendor;
- <AppName> - the name of a specific application;
- <AppVer> - the version of a specific application;
- <Packager> - the name of a packager.

Sample

This section provides a sample of an APS Catalog feed. For description of a specific feed element, refer to the Elements Reference section (on page 38).

```
<feed xmlns="http://www.w3.org/2005/Atom" xmlns:opensearch="http://a9.com/-
/spec/opensearch/1.1/">
  <id>http://apscatalog.com/1.atom</id>
  <title>{}</title>
  <updated>2008-04-07T10:35:34Z</updated>

  <!-- Derived from Opensearch 1.1 -->
  <opensearch:totalResults>73</opensearch:totalResults>
  <opensearch:startIndex>1</opensearch:startIndex>
  <opensearch:itemsPerPage>10</opensearch:itemsPerPage>
  <link rel="search" type="application/opensearchdescription+xml" href="" />
  <!-- Derived from RFC5005 -->
  <link rel="self" href="http://apscatalog.com/1.atom" />
  <link rel="first" href="http://apscatalog.com/1.atom" />
  <link rel="last" href="http://apscatalog.com/1.atom?page=8" />
  <link rel="next" href="http://apscatalog.com/1.atom?page=2" />
  <!-- Atom elements and APS Catalog extensions -->
  <entry xmlns:a="http://apstandard.com/ns/repo/1"
  xmlns="http://www.w3.org/2005/Atom">
    <!-- Package ID that conforms to RFC4287 and RFC4151 -->
    <id>tag:example.org,2008-02-27:/1/opensource/tikiwiki/1.9.7-
34/parallels</id>
    <!-- Package name. sa:name value -->
    <title>tikiwiki</title>

    <!-- Summary. sa:summary value -->
    <summary>tikiwiki</summary>

    <!-- The moment the package was added -->
    <updated>2003-12-13T18:30:02Z</updated>
    <!-- Application category. Multiple categories are permitted -->
    <category term="Web/Wiki" />
    <author a:author-type="vendor">
      <!-- Application vendor name -->
      <name>opensource</name>
      <!-- Application vendor URI -->
      <uri>http://info.tikiwiki.org/tiki-index.php</uri>
    </author>
    <author a:author-type="packager">
      <name>Parallels</name>
      <uri>urn:uuid:60a76c80-d399-11d9-b91C-0003939e0af6</uri>
    </author>
    <!-- Application description. sa:description content -->
    <content type="html">TikiWiki enable administrators and users to
create, display ...</content>
    <!-- URI of APS package bits -->
    <link rel="http://apstandard.com/ns/repo/1#bits"
type='application/zip'
      href="http://example.org/1/opensource/tikiwiki/1.9.7-
34.aps?packager=parallels" length="439144">
      <!-- any number of a:checksum elements is allowed -->
      <a:checksum algorithm="MD5">md5 digest</a:checksum>
      <a:checksum algorithm="SHA1">sha1 digest</a:checksum>
```

```

</link>
<!-- Link to Version index -->
<link rel="alternate"
href="http://example.org/1/opensource/tikiwiki/1.9.7-34/?packager=parallels"/>
<!-- APP-META link -->
<link rel="http://apstandard.com/ns/repo/1#meta"
type="application/xml"
href="http://example.org/1/opensource/tikiwiki/1.9.7-34.meta?packager=parallels"/>
<!-- Self link -->
<link rel="self" type="application/xml"
href="http://example.org/1/opensource/tikiwiki/1.9.7-34.atom?packager=parallels"/>
<!-- Direct screen shot links -->
<link rel="http://apstandard.com/ns/repo/1#screenshot"
type="image/png"
href="http://media.example.org/1/opensource/tikiwiki/1.9.7-34/resources/images/screenshot1.png" />
<!-- Link to Application resources index -->
<link rel="http://apstandard.com/ns/repo/1#resources"
type="application/xhtml+xml"
href="http://example.org/1/opensource/tikiwiki/1.9.7-34/resources/?packager=parallels"/>
<!-- Link to Package updates index. -->
<link rel="http://apstandard.com/ns/repo/1#updates"
type="application/atom+xml"
href="http://example.org/1/opensource/tikiwiki/1.9.7-34/updates.atom?packager=parallels"/>
<a:name>tikiwiki</a:name>
<a:version>1.9.7</a:version>
<a:release>34</a:release>
<a:description>TikiWiki enables administrators and users to create,
display ...</a:description>
<a:summary>tikiwiki</a:summary>
<a:vendor>opensource</a:vendor>
<a:packager>parallels</a:packager>
<a:categories>
  <a:category>Web/Wiki</a:category>
</a:categories>
</entry>
</feed>

```

Elements Reference

This section contains description of elements nested in an APS Catalog feed.

OpenSearch 1.1 elements nested in a feed:

Element	Description
//opensearch:totalResults	Total number of entries in a feed
//opensearch:startIndex	Index of the first entry in the current feed page
//opensearch:itemsPerPage	Number of entries per feed page
//link[@rel='search']	Link to a search description

[RFC5005] elements nested in a feed:

Element	Description
<code>//link[@rel='self']</code>	Link to the current feed page
<code>//link[@rel='first']</code>	Link to the first feed page
<code>//link[@rel='last']</code>	Link to the last feed page
<code>//link[@rel='next']</code>	Link to the next feed page

Other feed elements are divided into three groups:

- Native Atom elements that represent APS package metadata
- APS Catalog elements that represent APS package metadata
- APS Catalog service elements

For description of APS package metadata, see <http://apsstandard.com/r/doc/package-format-specification-1.0.pdf>.

Native Atom elements mapping:

Feed element	Package metadata element
<code>//id</code>	Package ID generated by APS Catalog
<code>//title</code>	<code>//sa:name</code>
<code>//summary</code>	<code>//sa:summary</code>
<code>//category/@term</code>	<code>//sa:categories/sa:category</code>
<code>//author[@a:author-type='vendor']/name</code>	<code>//sa:vendor/sa:name[not(@xml:lang)]</code>
<code>//author[@a:author-type='vendor']/uri</code>	<code>//sa:vendor/sa:homepage[not(@xml:lang)]</code>
<code>//author[@a:author-type='packager']/name</code>	<code>//sa:packager/sa:name[not(@xml:lang)]</code>
<code>//author[@a:author-type='packager']/uri</code>	<code>//sa:packager/sa:homepage[not(@xml:lang)]</code>
<code>//content</code>	<code>//sa:description[not(@xml:lang)]></code>
<code>//link[@rel='http://apstandard.com/ns/repol#icon']</code>	<code>//sa:icon</code>
<code>//link[@rel='http://apstandard.com/ns/repol#description']</code>	<code>//sa:screenshot</code>
<code>//link[@rel='http://apstandard.com/ns/repol#description']/a:description</code>	<code>//sa:screenshot/sa:description[not(@xml:lang)]</code>
<code>//a:name</code>	<code>//sa:name</code>
<code>//a:summary</code>	<code>//sa:summary[not(@xml:lang)]</code>
<code>//a:categories</code>	<code>//sa:categories</code>

//a:version	//sa:version
//a:release	//sa:release
//a:description	//sa:description[not(@xml:lang)]
//a:recommended	//sa:recommended
//a:upgradable-from	//sa:upgradable-from
//a:patchable-from	//sa:patchable-from
//a:patches	//sa:patches
//a:upgrades	//sa:upgrades

APS Catalog service elements:

Feed element	Description
//link[@rel='http://apstanda rd.com/ns/repo/1#bits']/a:ch ecksum	Package digest. Digesting algorithm is specified by the /entry/link/a:checksum/@algorithm attribute. Recommended algorithm values are SHA1 and MD5.
//link[@rel='http://apstanda rd.com/ns/repo/1#meta']/@hr ef	Link to package metadata
//link[@rel='http://apstanda rd.com/ns/repo/1#bits']/@hr ef	Link to package bits
//link[@rel='http://apstanda rd.com/ns/repo/1#updates']/@ href	Link to Package Updates feed
//link[@rel='http://apstanda rd.com/ns/repo/1#resources'] /@href	Link to Package Resources index
//link[@rel='http://apstanda rd.com/ns/repo/1#screenshot']/a:width	Screenshot width
//link[@rel='http://apstanda rd.com/ns/repo/1#screenshot']/a:height	Screenshot height
//link[@rel='alternate']/@hr ef	Link to index document
//link[@rel='http://apstanda rd.com/ns/repo/1#meta']/@hr ef	Link to package metadata (APP META)
//link[@rel='http://apstanda rd.com/ns/repo/1#certificate' ']/@href	Link to package certificate (if any)
//link[@rel='http://apstanda rd.com/ns/repo/1#certificate' ']/a:level	Optional tag that specifies package certification level. Possible values are <i>none</i> , <i>silver</i> , <i>gold</i> . Values are case insensitive. Missing or empty a:level tag is treated as <i>none</i> .

Search Descriptions

Each search description uses only standard OpenSearch 1.1 elements. The XML schema of a search description is available at <http://www.opensearch.org/Specifications/OpenSearch/1.1>.

Search Description URLs

Each search description is available at one of the following URLs:

Search description name	Feed URL
Repository	<ROOT_URL>/<APSver>.atom
Vendor	<ROOT_URL>/<APSver>/<VendorName>.atom
Application	<ROOT_URL>/<APSver>/<VendorName>/<AppName>.atom
Application Version	<ROOT_URL>/<APSver>/<VendorName>/<AppName>/<AppVer>.atom

The placeholder strings mean the following:

- `<ROOT_URL>` - the APS Catalog URL;
- `<APSver>` - the APS version. Allowed values: 1, 1.1, all;
- `<VendorName>` - the name of a specific vendor;
- `<AppName>` - the name of a specific application;
- `<AppVer>` - the version of a specific application.

Remarks

Package updates feed does not have linked search description, but you can also selectively fetch entries from this type of feed by adding certain common arguments to the feed request.

Search Description Arguments

Search descriptions define arguments that can be added to a feed to narrow a search results.

The following table lists arguments that can be added to feed requests.

Argument	Description	Type	Match	Example
<code>pageSize</code>	Number of entries per feed page	integer. Default value: 10	No	12
<code>page</code>	Feed page number	integer. Default value: 1	No	3
<code>order</code>	Comma-separated list of keys according to which feed entries within a feed must be ordered. "-" (dash) prefix specifies the descending order.	enum: <ul style="list-style-type: none"> ▪ name. Package name; ▪ packager. Packager name; ▪ vendor. Vendor name; ▪ updated. Last modification date. Default value: - updated	No	vendor,- name
<code>latest</code>	Limits the feed contents to the latest version of the package.	enum: 1	exact	1
<code>cert</code>	Certification level. If this argument is present in a feed request, APS Catalog returns details on packages of the specified certification level.	enum: <ul style="list-style-type: none"> ▪ None; ▪ Silver; ▪ Gold. 	greater or equal	Silver

vendor	Vendor ID	string	exact	www.phpbb.org
vendorSubstring	Limits the feed contents to those packages which vendor name starts with the given value.	string	substring	phpbb
category	APS Category ID.	string	exact	Web/Wiki
categorySubstring	Limits the feed contents to those packages which category name starts with the given value. Conventional category names are listed in the http://www.apsstandard.com/r/doc/aps--application-categories.pdf document, but a client should accept other category names.	string	substring	Web
name	Case-insensitive package commercial name. It is allowed to use more than one name argument in feed URL.	string	substring	PHPbb
packager	Packager name	string	exact	www.example.com
obsolete	A package is considered obsolete if there is another package in the same catalog which upgrades the package. By specifying "hide" for this parameter you exclude obsolete packages from the requested feed.	string	exact	hide
arch	Target architecture	enum: <ul style="list-style-type: none"> ▪ undefined; ▪ x86; ▪ x86_64. Default value: undefined.	exact	x86
package_type	Package type.	enum: <ul style="list-style-type: none"> ▪ aps; ▪ pvc. Default value: aps	exact	aps
platform	Case-insensitive target platform	enum: <ul style="list-style-type: none"> ▪ undefined; ▪ linux; ▪ windows; ▪ freebsd; ▪ macos. Default value: undefined.	exact	linux
os	Case-insensitive target operating system	enum. Default value: undefined	exact	undefined

Restrictions

- *Vendor* feed do not support `vendor` and `vendorSubstring` arguments as `vendor` is already specified in feed URL.
- *Package* feed do not support `vendor` and `name` arguments as these parameters are already specified in feed URL.

Comparison rules

The *Match* column defines comparison rules which APS Catalog uses to identify that there is a match between an argument value and entry data. In case of match, the entry is returned. The rules are as follows:

- *Exact*. The value of a specific entry element must be equal to an argument value.
- *Substring*. The value of a specific entry element must include an argument value.
- *No*. The argument is not compared with entry data.

Examples

- `<ROOT_URL>/<APSver>/<VendorName>.atom?page=3`
- `<ROOT_URL>/<APSver>.atom?name=php&name=Drupal`

CHAPTER 8

Operations on Content Resources

Content resource is an APS Catalog resource that is not a feed, index, or search description. The following types of content resources are available:

- Package bits
- Package resources

Package metadata You can perform the following operations on content resources:

- **ADD** operation. This operation adds a resource to APS Catalog by using HTTP POST request;

Note: Package bits, package metadata, and package resources are added to APS Catalog only on adding a package.

- **RETRIEVE** operation. This operation retrieves a resource using HTTP GET request;

This section describes request arguments and possible response codes for these operations.

In This Chapter

ADD Package.....	45
RETRIEVE Operations.....	47

ADD Package

Two methods of adding content resources to APS Catalog are available:

- using APS Catalog form;
- using Packager Administration service.

APS Catalog Form Arguments

Action URL

This URL is fetched from a certain Repository index. XPath expression:
`/html/body/form[@id='packageSubmitForm']/@action/text()`.

Request Arguments

Argument name	Description	Type	Required
package	Package bits	file	yes(*)
meta	Package meta	file	yes(*)
url	Package bits URL	text	yes(*)

commercialName	Package commercial name	text	no
packageType	Package type (APS for standard APS package, PVT for container template packaged as APS)	enum. Allowed values: APS PVT	yes

* - Either package or both meta and url are required.

Response Codes

Code	Description
201 Created	The package has been successfully uploaded to the repository. The Location header indicates the URL of the newly created Package index document (see Package Index).
400 Bad request	The submitted package did not pass validation and cannot be accepted. The body of the response may contain detailed reason of failure.
401 Unauthorized	Authorization required. The WWW-Authenticate header will specify the expected authentication scheme.
403 Forbidden	The credentials specified are not valid for this operation.
405 Method not allowed	The handler does not accept the request method. Only GET and POST methods are accepted.

Packager Administration Panel Request Arguments

Action URL

<ROOT_URL>/packager/upload/

Request Arguments

Argument name	Description	Type	Required
aps_upload_param_prefix	Prefix for upload parameters, could be empty	string	yes(*)
<PREFIX>username	Packager login	string	yes(*)
<PREFIX>password	Packager password	string	yes(*)
<PREFIX>approve	Request for automatic package approval; packager should have permission to complete automatic approval	boolean	yes(*)
<PREFIX>publish	Request for automatic publishing (enabling) of new package; packager should have permission to complete automatic publishing.	boolean	yes(*)
<PREFIX>commercialName	Package commercial name	string	no

<PREFIX>file	package archive itself	file	yes
--------------	------------------------	------	-----

Response Codes

Code	Description
200 Ok	The package has been successfully uploaded to the repository.
400 Bad request	The submitted request did not pass validation and cannot be accepted. The body of the response may contain detailed reason of failure.
401 Unauthorized	Authorization failed, provided credentials are not valid. The WWW-Authenticate header will specify the expected authentication scheme.
403 Forbidden	The credentials specified are not valid for this operation.
405 Method not allowed	The handler does not accept the request method. Only POST method is accepted.
500 Server Error	The submitted package did not pass validation with APSLint and cannot be accepted; or the submitted package already exists in APS Catalog; or an internal framework error occurred.

RETRIEVE Operations

Resource locations

Package bits URL: / <APSver> / <VendorName> / <AppName> / <AppVer> .aps

Package metadata URL: / <APSver> / <VendorName> / <AppName> / <AppVer> .meta

Package certificate URL:

/ <APSver> / <VendorName> / <AppName> / <AppVer> .certificate

The placeholder strings have the following meaning:

- <ROOT_URL> - the APS Catalog URL
- <APSver> - the APS version. Allowed values: 1, 1.1, all
- <VendorName> - the name of a specific vendor
- <AppName> - the name of a specific application
- <AppVer> - the version of a specific application

Request Arguments

Argument name	Description	Type	Required	Example
packager	Packager name	string	yes	www.example.com
arch	Target architecture	enum**. Default value: undefined	no	x86

type	Package type. One of PVT or APS	enum**. Default value: APS	no	APS
platform	Case-insensitive target platform	enum**. Default value: undefined	no	linux
os	Case-insensitive target operating system	enum**. Default value: undefined	no	undefined

Response Codes

Code	Description
300 Multiple Choices	Multiple resources for this request. Client should retrieve a list of available resources (either through feed or index resources) and choose the appropriate one.
302 Found 303 See Other	Resource is available at other URL. Typically, this is used to redirect a client to an URL which includes all request arguments with specific values. The client must follow the URL specified in Location header.
404 Not Found	Requested resource is not found.

Index

A

About APS Catalog • 4
About This Guide • 5
ADD Package • 48
Adding Content to APS Catalog • 22
Adding Package to Catalog Directly • 22
Adding Package via Packager Administration Panel • 23
Advanced Browsing • 15
Application Index • 32
Application Version Index • 34
APS Catalog API Operations • 12
APS Catalog API Overview • 8
APS Catalog API Reference • 24
APS Catalog Form Arguments • 48
APS Catalog Resources • 9

B

Browsing APS Catalog • 13

D

Discovering Feeds • 19
Discovering Repository Feed • 14

E

Elements Reference • 41

F

Feed URLs • 38
Feedback • 7
Feeds • 38
Filters • 16

I

Index URLs • 25
Indexes • 25
Introduction • 4

O

Operations on Content Resources • 48

P

Package Index • 36
Package Resources Index • 37

Packager Administration Panel Request Arguments • 49

R

Repository Index • 27
RETRIEVE Operations • 50
Retrieving Information from Indexes • 20
Retrieving Repository Feed and Displaying Its Content • 14
Retrieving Repository Index • 13

S

Sample • 39
Search Description Arguments • 45
Search Description URLs • 44
Search Descriptions • 18, 44
Service Index • 26
Service Operations • 18

T

Technical Details • 8
Typographical Conventions • 6

U

Useful Links • 6

V

Vendor Index • 30