DIGARC
curriculum + catalog

Acalog Web Services

USAGE MANUAL V1.1

Revision date: 4/2016

Table of Contents

1. API Rules and Conditions

This Schedule sets forth the terms on which Digarc makes the Acalog API (as defined below) available for use by approved licensees of Acalog. Access to and use of the Acalog API requires that a duly authorized representative of the Institution agree to be bound by the following API Rules and Conditions.

ACALOG API

The Acalog API provides the Institution with the ability to search and retrieve structured (XML formatted) public and private academic catalog content from Acalog in order to build external services ("Applications") that rely on catalog content stored in Acalog.

The API is provided as a REST-based Web service, which is secured by use of either a private or public key, which may be generated and obtained from workflow in the Acalog Publisher. The private key permits access to all catalog content, including content that may otherwise be set to a status of "inactive" or not "published". The public key limits access to content that is published and available through the Institution's Acalog e-catalog Gateway.

RULES AND CONDITIONS

1. Institution may use the Acalog API to develop Applications to search, retrieve, view, and display catalog content anywhere within its institutional Website, and only under Institution's *.edu domain. Institution is entirely responsible for the development of said Applications; and for security, maintenance, troubleshooting, performance, availability, usability, function, form, and applicability of use.
2. Institution may permit its staff, faculty, researchers, students, and prospective students to use the Applications developed using the API for academic and educational purposes only.
3. Institution may use the API to develop Applications to integrate catalog content from Acalog with other campus systems, but may not provide API documentation ("Documentation"), or public or private API keys, to any third-party without the express written consent of Digarc.
4. Digarc shall have the right to use and adapt Institution's public catalog content available through the API in order to provide enhanced services to the Institution, to its customers, and to the general public.
5. Requests to the API by the Institution's Applications should not exceed 150 per hour. This rate limit is put into place to prevent excessive use of the API. Best practices should be observed to avoid exceeding this limit. They are as follows:
   a. Caching. Store API responses in your Application or on your Website if you expect high-volume usage. For example, avoid calling the API on every page load of your Website. Instead, call the API infrequently, cache the response on your end, and display the local version on page loads.

b. Prioritize and limit highly interactive hits. For example if you have a custom search that utilizes the API, implement your own limits such as total number of searches allowed by IP or preventing the user from performing a search more than once every 10 seconds. Caching search results is also a good idea.

6. You may apply to have the rate limit referenced above increased, by which your Application will be "whitelisted". Depending on use, the Institution may be required to pay an additional maintenance fee as compensation for additional bandwidth or system resources associated with use of the API beyond the rate limit. A request for whitelisting must be submitted via the Acalog Publisher Support module, and should include the following:

   a. Describe your application in detail:
   b. Will you be developing this in-house or is this for an external project:
   c. Approximate number of hits per hour to the API:
   d. Contact information for the project manager and lead developer for this application:

7. You may apply for permission to provide your public key to a third-party for use outside of your *.edu domain. Such use may not violate the terms of this Schedule, or of the Software License and Hosting Agreement. The third-party may be required to execute a non-compete, non-disclosure agreement related to use of the API and related documentation. Depending on use, third-party or the Institution may be required to pay an additional maintenance fee as compensation for additional bandwidth or system resources associated with use of the API. A request for such use must be submitted via the Acalog Publisher Support module, and should include the following:

   a. name and address of third-party:
   b. Planned use of the API by the third-party:
   c. Approximate number of hits per hour to the API:
   d. Contact information for business and technical representatives of the third- party:

8. Institution may not separately extract and provide or otherwise use data elements from the API to enhance the data files of third parties.

9. Institution further agrees not to otherwise reproduce, modify, distribute, decompile, disassemble or reverse engineer any portion of the API or API documentation.

10. If Institution becomes aware that any user of the API or Institution-developed Applications is in material breach of these API Rules and Conditions, Institution agrees to notify Digarc promptly of such breach by e-mail to support@digarc.com, and to take prompt corrective action at its expense to remedy such breach.

11. Digarc reserves the right, at its sole discretion, to disable use of and access to the API for failure to follow these Rules and Conditions. Digarc will, however, make a good-faith attempt to provide notice of breach, and provide reasonable time for correction prior to taking action to disable access to the API.

12. The parties may execute more than one counterpart of this Agreement hereto, and each fully executed counterpart shall be deemed an original.

2. Conventions Used In This Document

We have attempted to make this manual as easy to read and navigate as possible. To help, a few basic conventions will be used throughout.

Each of this manual's sections may contain several sub-sections, which, in turn, may have their own sub-sections and so on. All sections use an outline number format to help you quickly locate information.

All code is formatted using `Consolas` Italics font. PHP and XML examples include syntax highlighting to make them easier to read.

URL argument values are formatted using `Consolas`, and enclosed in greater than, and less than signs indicating value type, or example values: `<int>`

Optional URL arguments are formatted using italics and surrounded by square brackets. For example:

https://apis.acalog.com/v1/content?format=xml&key=`<key>`&method=getCatalogs*[&only_active=<boolean>]*

Within examples, portions specific to the above text are highlighted in **bold**.

Any important notes or shortcuts that relate to a section will be indented, shaded and in italics. For example,

> ***Note:*** *this is an example note*

Some places in this manual will refer to other sections for additional information. Such references will be *underlined and italicized*.

> ***Note:*** *examples may be modified for formatting (i.e. XML declaration and namespace declarations may be removed).*

> ***Note:*** *for brevity, most samples will have duplicated structure truncated.*

3. Getting Started

The Acalog Web services platform consists of several services. This document covers all aspects of each of these services.

Current services are:

- Catalog Search & Listing
- Content Retrieval

To use the API, you will need an API key. A key can be created or retrieved using the Remote Services module in the Acalog Publisher.

3.1 API KEYS

There are two types of keys, Public and Private. Public keys are used to retrieve publically published catalog content. Private keys can additionally retrieve non-publically published catalog content (i.e. from unpublished catalogs, or inactive items in published catalogs). All keys are read-only.

Public Key: Access to publically available content

Private Key: Access to publically and non-publically available content

> ***Note:*** *The responses from the web service differ when using a private or public key. When using a private key, the differences are additional meta-data elements (i.e. creation/modification date) in addition to all public data.*

3.2 USING THE APIS

The Acalog Web services platform utilizes REST XML Web services over HTTP.

Using standard HTTP GET or POST requests, any HTTP client can interact with the services. All responses are standard, valid XML responses that can be parsed using any XML toolkit.

All data is considered to be string data, therefore you may use any programming language to interact with the Web services.

These services were written using PHP, and utilize some advanced XML standards such as xinclude; natively supported by libxml2 and exposed through PHP. Some XML toolkits may not provide this ability and therefore the xincludes must be parsed programmatically.

Xincludes are used to represent the relational aspects of the data, to remove repetition of content within the XML documents that would otherwise make the documents much larger, and the HTTP requests much slower.

All documents are sent as UTF-8 (Unicode). All requests must be sent as UTF-8 (Unicode).

3.3 ACCESSING THE SERVICE

All services are accessed via **https://apis.acalog.com** (for self-hosted clients, this will be different).

To access services, you must pass in several arguments as part of the resource URL (this determines the service and version of the service requested), as well as query arguments.

3.3.1 Path Arguments

The order of path arguments must be as listed below.

3.3.1.1 API Version

The first argument is the API version; this is passed in the PATH of the URL.

Example: https://apis.acalog.com/**v1**/

3.3.1.2 Service Name

The second argument is the Service name; this is also passed in the PATH of the URL.

Example: https://apis.acalog.com/v1/**search** or https://apis.acalog.com/v1/**content**

3.3.2 Query Arguments

Query arguments may be sent in any order, and included in either GET or POST data. All query arguments except arrays are defined using the standard key-value pair format, e.g. *key=value*.

Arrays are defined using square brackets to denote the index, which may be numeric (zero-based integer) or string types, e.g. *key[index]=value*. It is also possible to use anonymous indexes if you are using contiguous numeric indexes. This means *key[]=value&key[]=value2* is identical to *key[0]=value&key[1]=value2*. You may also nest arrays using the format *key[index][index]=value*.

3.3.2.1 Argument Types

- **Boolean**

  Boolean arguments must be passed as a 1 (true) or 0 (false)

- **String**

  String arguments are simple unicode-compliant string values.

- **Integer**

  Integer arguments are always positive, whole numbers.

- **Array**

  Array arguments may contain any other type of value, including other arrays.

3.3.2.2 Required Arguments

3.3.2.2.1 API Key

- Required: *for all requests*
- Name: *key*
- Value: A valid API key

Example: https://apis.acalog.com/v1/content?key=*<key>*

3.3.2.2.2 Format

- Required: *for all requests*
- Name: *format*
- Value: *xml*

Example: https://apis.acalog.com/v1/content?key=<key>&catalog=*<id>***&format=*xml***

3.3.2.2.3 Catalog ID

- Required: *unless stated*
- Name: *catalog*
- Value: *integer*

Example: https://apis.acalog.com/v1/content?key=*<key>***&catalog**=*<id>*

3.3.2.2.4 Organization ID (Enterprise Edition Only)

- Required: *unless stated*
- Name: *organization*
- Value: *integer*

Example: https://apis.acalog.com/v1/content?key=*<key>*&catalog=*<id>***&organization**=*<id>*

4. Using the Content Service

The Content service provides access to full item content based on its ID. It is intended to be used in conjunction with the search service, and items are retrieved using the IDs returned within search results.

In addition, the content service can be used to retrieve information about accessible catalogs, organizations (enterprise edition only), hierarchy items, types and course prefixes.

4.1 ACCESSING THE CONTENT SERVICE

The content service can be accessed via https://apis.acalog.com/v1/**content**

4.2 METHODS

To call a method, pass in the method query argument with the desired method name for its value. Appropriate arguments are listed below.

Example: https://apis.acalog.com/v1/content?format=xml&key=*<key>***&**method=*<method>*

4.2.1 Get Organization List

The `getOrganizations` method allows you to retrieve a list of organizations. *This method is available with Acalog Enterprise Edition only.*

4.2.1.1 Query Arguments

This method has no additional arguments. *This method does not require the organization ID.*

4.2.1.2 Example URI

https://apis.acalog.com/v1/content?format=xml&key=*<key>***&method=getOrganizations**

4.2.1.3 Example Response

```
<organizations>
    <organization id="acalog-organization-1">
        <a:title>Graduate School</a:title>
    </organization>
</organizations>
```

4.2.2 Get Catalogs List

The `getCatalogs` method allows you to retrieve a list of catalogs. Catalogs are sorted in the order set in the Acalog Publisher.

4.2.2.1 Query Arguments

This method takes a single argument. *This method does not require the catalog ID*.

Show Only Active Catalogs

- Required: *no*
- Name: *only_active*
- Type: *boolean*
- Default: *true*

4.2.2.2 Example URI

https://apis.acalog.com/v1/content?format=xml&key=*<key>**&method=getCatalogs[&only_active=<boolean>]***

4.2.2.3 Example Response

```
<catalogs>
    <types>
        <catalog>
        <type id="acalog-catalog-type-1">
            <a:title>Undergraduate</a:title>
            <a:content visibility="hidden" />
            <state>
            <code>0</code>
            <name>Inactive-Hidden</name>
            </state>
            <created>2014-01-13 15:50:47</created>
            <modified>2014-01-13 15:50:47</modified>
        </type>
        ...
        </catalog>
    </types>
    <catalog id="acalog-catalog-1">
        <a:title>2013-2014 Undergraduate Catalog</a:title>
        <comments />
        <created>09/27/2014 13:38:25</created>
        <modified>03/23/2014 17:26:10</modified>
        <state>
        <published>Yes</published>
        <archived>No</archived>
        </state>
        <type>
        <xi:include xi:xpointer="xmlns(c=http://acalog.com/catalog/1.0)
        xpointer((//c:types//c:type[@id='acalog-catalog-type-1'])[1])">
            <xi:fallback>undergrad</xi:fallback>
        </xi:include>
        </type>
    </catalog>
    ...
    <status>success</status>
</catalogs>
```

4.2.3 Get Hierarchy List

The *getHierarchy* method allows you to retrieve the entire hierarchy for a single catalog. The XML response contains a legend to identify different types of hierarchy items (i.e. School/College or Departments), and within that, the customized term used by the school is given as the <*localname*>.

4.2.3.1 Query Arguments

This method has no additional arguments.

4.2.3.2 Example URI

https://apis.acalog.com/v1/content?format=xml&key=<*key*>&catalog=<*id*>&**method**=**getHierarchy**

4.2.3.3 Example Response

```
<catalog id="acalog-catalog-6">
```

```
<hierarchy>
    <legend>
    <key id="acalog-entity-type-7">
        <name>Institution</name>
        <localname>Institution</localname>
    </key>
    <key id="acalog-entity-type-6">
        <name>College</name>
        <localname>School/College</localname>
    </key>
    <key id="acalog-entity-type-5">
        <name>Department</name>
        <localname>Department</localname>
    </key>
</legend>
<entity id="acalog-entity-139">
    <type>
        <xi:include xi:xpointer="xmlns(c=http://acalog.com/catalog/1.0)
        xpointer(//c:key[@id='acalog-entity-type-7'])"/>
    </type>
    <a:title>The University</a:title>
    <a:content>
        <h:p>Please see navigational links to the left for institutional
        information.</h:p>
    </a:content>
    <state>
    <code>1</code>
    <name>Active-Visible</name>
    </state>
    <children>
        <entity id="acalog-entity-140">
        ...
    </children>
```

```
        </entity>
    </hierarchy>
    <status>success</status>
</catalog>
```

4.2.4 Get Items

The `getItems` method allows you to retrieve the full content for one or more items.

> *This method will default to returning abbreviated item content when multiple items are requested. To change this behavior pass in the `full` option.*

4.2.4.1 Query Arguments

This method takes two arguments.

Item Type

- Required: *yes*
- Name: *type*
- Type: *string*
- Default: *none*
- Possible values: *programs, courses, catalogs, hierarchy, pages*

Item IDs

- Required: *yes*
- Name: *ids*
- Type: *array*
- Values:
    - *valid integer item IDs*
        - Default: *none*

Options

- Required: *no*
- Name: *options*
- Type: *array*
    - *full*
        - Type: ***integer***
        - Default: ***0***
        - Values:
            - ***0*** (return abbreviated item details)
            - *1* (return full item contents)

4.2.4.2 Example URI

https://apis.acalog.com/v1/content?format=xml&key=<*key*>&catalog=<*id*>**&method=getItems&type=**<*type*>**&ids[]=**<*int*>***[&ids[]=***<*int*>***]***

4.2.4.3 Example Response

```
<catalog id="acalog-catalog-6">

    <courses>

        <fields>

            <field id="acalog-field-60" template="title" type="small-text"  required="yes">

                <a:title>Prefix</a:title>

            </field>

            <field id="acalog-field-67" template="title" type="space">

                <a:title> </a:title>

            </field>

            <field id="acalog-field-74" template="title" type="small-text" required="yes">

                <a:title>Code</a:title>

            </field>

            <field id="acalog-field-81" template="title" type="static-text">

                <a:title> - </a:title>

            </field>

            <field id="acalog-field-88" template="title" type="small-text"  required="yes">

                <a:title>Name</a:title>

            </field>

            <field id="acalog-field-43" template="body" type="horizontal-rule"
            visibility="visible">

                <a:title> </a:title>

                <display>

                    <name visibility="visible" bold="no" italic="no" linebreak="no"
                    placement="before"></name>

                    <data bold="no" italic="no"></data>

                </display>

            </field>

        </fields>

        <course id="acalog-course-6058">

            <a:title>AEDU 201 - Introduction to Visual Arts Education</a:title>

            <parent>

                <xi:include xi:xpointer="xmlns(c=http://acalog.com/catalog/1.0)
                xpointer((//c:hierarchy//c:entity[@id='acalog-entity-341'])[1])">
                    <xi:fallback>Division of Business and Communication</xi:fallback>
                </xi:include>
            </parent>
            <type>
                <xi:include xi:xpointer="xmlns(c=http://acalog.com/catalog/1.0)
                xpointer(//c:types//c:type[@id='acalog-course-type-306'])">
                    <xi:fallback>Art Education</xi:fallback>
                </xi:include>
            </type>
            <state>
                <code>1</code>
                <name>Active-Visible</name>
                <parent>
                    <code>1</code>
                    <name>Active-Visible</name>
                </parent>
            </state>
            <created>2008-09-16 11:14:58</created>
            <modified>2009-07-28 14:45:24</modified>
            <a:content>
                <prefix>AEDU</prefix>
                <code>201</code>
                <name>Introduction to Visual Arts Education</name>
                <field type="acalog-field-60">
                    <xi:include xi:xpointer="xmlns(c=http://acalog.com/catalog/1.0)
                    xpointer(//c:courses/c:fields/c:field[@id='acalog-field-
                    60'])"></xi:include>
                    <data>AEDU</data>
                </field>
                ...
            </a:content>
        </course>
        ...
    </courses>
```

```
    <types>
        <course>
            <type id="acalog-course-type-306">
                <a:title>Art Education</a:title>
                <a:content visibility="hidden"></a:content>
            </type>
        </course>
    </types>
    <status>success</status>
</catalog>
```

4.2.5 Get Course Prefix List

The `getPrefixes` method allows you to retrieve a list of course prefixes within a single catalog.

4.2.5.1 Query Arguments

This method takes a single argument.

Show Only Active Course Prefixes

- Required: *no*
- Name: *only_active*
- Type: *boolean*
- Default: *true (1)*

4.2.5.2 Example URI

https://apis.acalog.com/v1/content?format=xml&key=<*key*>&catalog=<*id*>&**method**=**getPrefixes*[&only_active*=<**boolean**>]***

4.2.5.3 Example Response

```
<catalog id="acalog-catalog-6">
    <prefix>AEDU</prefix>
    <prefix>AETH</prefix>
    <prefix>CAIN</prgefix>
    <prefix>CMAC</prefix>
    <prefix>CMMC</prefix>
    <prefix>CRCM</prefix>
    ...
</catalog>
```

4.2.6 Get Table of Contents

The `getToc` method allows you to retrieve the table of contents for a single catalog. The table of contents is those items included in the gateway navigation.

4.2.6.1 Example URI

https://apis.acalog.com/v1/content?format=xml&key=<*key*>&catalog=<*id*>&**method**=**getToc**

4.2.6.2 Query Arguments

This method has no additional arguments.

4.2.6.3 Example Response

```
<catalog id="acalog-catalog-6">
    <toc>
        <link>
            <a:title>Catalog  Home</a:title>
            <xi:include xi:xpointer="xmlns(c=http://acalog.com/catalog/1.0)
            xpointer(//c:page[@id='acalog-page-275'])"/>
        </link>
        <link>
            <a:title>Colleges</a:title>
            <xi:include xi:xpointer="xmlns(c=http://acalog.com/catalog/1.0)
            xpointer(//c:page[@id='acalog-page-310'])"/>
        </link>
        <h:br/>
        <link>
            <a:title>General Education Requirements</a:title>
            <xi:include xi:xpointer="xmlns(c=http://acalog.com/catalog/1.0)
            xpointer(//c:program[@id='acalog-program-485'])"/>
        </link>
        <link>
            <a:title>2009-10 four year academic calendar (PDF)</a:title>
            <xi:include xi:xpointer="xmlns(c=http://acalog.com/catalog/1.0)
            xpointer(//c:file[@id='acalog-media-309'])"/>
        </link>
        ...
    </toc>
    <pages>
        <page id="acalog-page-275">
            <a:title>Catalog Home</a:title>
            <linkname>Catalog Home</linkname>
        <state>
            <code>1</code>
            <name>Active-Visible</name>
        </state>
    </page>
    <page id="acalog-page-310">
            <a:title>Colleges</a:title>
            <linkname>Colleges</linkname>
            <template>Schools/Colleges</template>
            <state>
                <code>1</code>
                <name>Active-Visible</name>
            </state>
        </page>
        ...
    </pages>
    <media>
        <file id="acalog-media-309">
            <a:title>pdf of calendar</a:title>
            <a:link type="application/pdf"
            href="http://catalog.uarts.edu/mime/media/view/7/309/calendar%204yr%2009-
            13%20FINAL.pdf"/>
            <state>
                <code>1</code>
                <name>Active-Visible</name>
            </state>
        </file>
        ...
    </media>
    <programs>
```

```
        <program id="acalog-program-485">
            <type>
                <xi:include xi:xpointer="xmlns(c=http://acalog.com/catalog/1.0)
                xpointer(//c:types//c:type[@id='acalog-program-type-52'])">
                    <xi:fallback>Bachelor's Degree Programs</xi:fallback>
                </xi:include>
            </type>
            <type>
                <xi:include xi:xpointer="xmlns(c=http://acalog.com/catalog/1.0)
                xpointer(//c:types//c:type[@id='acalog-degree-type-73'])">
                    <xi:fallback>Bachelor of Fine Arts</xi:fallback>
                </xi:include>
            </type>
            <a:title>General Education Requirements</a:title>
        </program>
        ...
    </programs>
    <types>
        <program>
            <type id="acalog-program-type-52">
                <a:title>Bachelor's Degree Programs</a:title>
                <a:content visibility="hidden"/>
            </type>
            <type id="acalog-degree-type-73">
                <a:title>Bachelor of Fine Arts</a:title>
                <a:content visibility="hidden"/>
            </type>
            ...
        </program>
        ...
    </types>
</catalog>
```

4.2.7 Get Item Types

The `getItems` method allows you to retrieve the list of types for a given item type in a single catalog. You may also get a list of catalog types, and must omit the catalog argument.

4.2.7.1 Query Arguments

This method takes one arguments.

Item Type

- Required: *yes*
- Name: *type*
- Type: *string*
- Default: *none*
- Possible values: *programs, degrees, courses, catalogs, schools, colleges, departments*

Options

- Required: *no*
- Name: *options*
- Type: *array*
    - *status*
        - Type: *string*
        - Default: *active and active-hidden*

- **Values:**

- active
- active-hidden
- inactive

4.2.7.2 Example URI

https://apis.acalog.com/v1/content?format=xml&key=<*key*>&catalog=<*id*>**&method**=**getTypes&**

type=<*type*>

4.2.7.3 Example Response

```
<catalog id="acalog-catalog-6">
    <types>
        <course>
            <type id="acalog-course-type-306">
                <a:title>Art Education</a:title>
                <a:content visibility="hidden">
                    <h:p>Art Education</h:p>
                </a:content>
                <state>
                    <code>1</code>
                    <name>Active-Visible</name>
                </state>
                <created>09/27/2013 13:38:25>
                <modified>09/27/2013 13:38:25>
            </type>
            ...
            <type id="acalog-course-type-308">
                <a:title>College of Performing Arts College-Wide Offering</a:title>
                <a:content visibility="hidden" />
                <state>
                    <code>1</code>
                    <name>Active-Visible</name>
                </state>
                <created>09/27/2013 13:38:25>
                <modified>09/27/2013 13:38:25>
            </type>
            ...
        </course>
    </types>
    <status>success</status>
</catalog>
```

4.3 PERMALINKS

Within Acalog it is possible to create dynamic links between [almost] any two items in the system. These links automatically update, keeping the relationship intact when catalogs are rolled over and changes to the linked data (i.e. its name) are made. These are called “Permalinks”.

Within the XML documents returned via the content service, these permalinks are represented as `<permalink>` elements. They contain all of the information required to re-create the permalink as it appears within Acalog. In addition, the item being linked is included within the document in abbreviated form so that in the majority of cases (the name is the text of the link) you may reproduce the link without requiring a second request to the web service.

4.3.1 Example

Within the Prerequisites for a course, you might include permalinks to required courses:

```
<field type="acalog-field-46">
    <xi:include xi:xpointer="xmlns(c=http://acalog.com/catalog/1.0)
    xpointer(//c:courses/c:fields/c:field[@id='acalog-field-46'])"/>
    <data>
        <h:p>Priority enrollment to Art Education majors and concentrations.</h:p>
        <h:p>
            The following courses:
            <permalink link-id="acalog-permalink-505" to="acalog-course-1727" type="dynamic">
                <a:title>AEDU 201</a:title>
            </permalink>
            <permalink link-id="acalog-permalink-685" to="acalog-course-2150" type="dynamic">
                <a:title>AEDU 499</a:title>
            </permalink>
        </h:p>
    </data>
</field>
```

The permalink is comprised of an `<a:title>` element, and an XInclude. The `<a:title>` element contains the text of the link that would appear on the catalog gateway. The XInclude resolves to the abbreviated version of the item, for example:

```
<course id='acalog-course-6058'>
    <a:title>AEDU 201 - Introduction to Visual Arts Education</a:title>
    <type>
        <xi:include xi:xpointer="xmlns(c=http://acalog.com/catalog/1.0)
        xpointer(//c:types//c:type[@id='acalog-course-type-306'])">
            <xi:fallback>Art Education</xi:fallback>
        </xi:include>
    </type>
</course>
<course id='acalog-course-6059'>
    <a:title>AEDU 499 - Internship</a:title>
    <type>
        <xi:include xi:xpointer="xmlns(c=http://acalog.com/catalog/1.0)
        xpointer(//c:types//c:type[@id='acalog-course-type-306'])">
            <xi:fallback>Art Education</xi:fallback>
        </xi:include>
    </type>
</course>
```

5. Using the Search Service

The Search service provides full-text searches against catalog data stored in the Acalog database.

5.1 ACCESSING THE SEARCH SERVICE

The search service can be accessed via https://apis.acalog.com/v1/**search/<*location*>**

The search service requires an additional PATH parameter, *location*, to identify the items in the catalog to be searched.

5.1.1 Search Locations

- Courses (https://apis.acalog.com/v1/search/**courses**)
- Programs (https://apis. acalog.com/v1/search/**programs**)
- Hierarchy Items (https://apis. acalog.com/v1/search/**hierarchy**)
- Pages (https://apis. acalog.com/v1/search/**pages**)

5.2 SEARCH RESULTS

Search results are comprised of an item ID, name, a relevance score and a state. Some items return other data in addition.

Courses

- Alternate Name (containing the Acalog full course title)
- Prefix
- Code
- Type (if available)

Programs

- Program Type (if available)
- Degree Type (if available)

Pages

- Alternate Name (may be the same as the name)

5.2.1 Example Response

```
<catalog>
    <search>
        <hits>52</hits>
        <results>
            <result>
                <id>450</id>
                <name>Liberal Arts Curriculum</name>
                <altname/>
                <type id="48">Programs of Study Pursued Within the UG degree</type>
                <state>
                    <code>1</code>
                    <name>Active-Visible</name>
                    <parent>
                        <code>1</code>
                        <name>Active-Visible</name>
                    </parent>
                </state>
            </result>
            ...
        </results>
    </search>
    <status>success</status>
</catalog>
```

5.3 QUERY SYNTAX

The search service has a comprehensive query syntax, allowing you to perform complex searches. The query should be passed in as a single string value for the `query` argument.

5.3.1 Keywords

The simplest form of search is using keywords — any document that matches one or more keywords will be returned.

Example: `Biology Sciences`

5.3.2 Phrases

To match an exact phrase — a combination of multiple keywords in a specific order, simply encase it in double quotes. This must also be done for any column filter values containing more than one keyword.

Example: *`"Introduction to Biology"`*

5.3.3 Boolean logic

AND/OR may be used to specify how multiple keywords are handled. Boolean operators **must** be uppercase.

Example: *`Math AND Chemistry OR Biology`*

You may also use parenthesis to indicate order-of-operations when using booleans operators.

Example: *`(Math AND prefix:MAT) OR (Math AND prefix:PHYS)`*

Note: there is an implicit *`OR`* applied if no operator is supplied.

5.3.4 Column Filters

Column filters allow you to filter results using item attributes. These can be used in addition to, or instead of, keywords and phrases. These use the format *"`columnname:value`"*. If the value consists of multiple keywords, you must quote them.

5.3.4.1 Available Courses Column Filters

- Prefix *(`prefix`:ANTH)*
- Code *(`code:101`)*
- Parent *(`parent:"Department of Humanities"`)*
- Type *(`type:Science`)*
- Custom Fields *(`prerequisites:"BIO 120"`)*

- The column name is the field name with all non-alphanumeric characters removed

(e.g. "Prerequisites & Co-Requisites" betcomes *`prerequisitescorequisites`*)

Both parent and type column filters accept a name or numeric ID.

5.3.4.2 Available Programs Column Filters

- Parent *(`parent:"College of Arts"`)*
- Program Type *(`type:Math`)*
- Degree Type *(`degreetype:"Bachelor of Science"`)*

Both parent and type column filters accept a name or numeric ID.

5.3.4.3 Available Hierarchy Column Filters

- Parent *(`parent:"College of Arts"`)*
- Type *(`type:"Language Studies"`)*

Both parent and type column filters accept a name or numeric ID.

5.4 METHODS

To call a method, pass in the method query argument with the desired method name for its value.

Example: https://apis.acalog.com/v1/search/*<location>?*__method=<method>__

5.4.1 Search

The *search* method allows you to retrieve items from the Acalog database based on a search query, as documented above.

5.4.1.1 Query Arguments

This method takes two arguments.

Search Query

- Required: *yes*
- Name: *query*
- Type: *string*

Options

- Required: *no*
- Name: *options*
- Type: *array*
    - *sort*
        - Type: *string*
        - Default: *alpha*
        - Values:
            - *alpha*
            - *rank*
    - *limit*
        - Type: *integer*
        - Default: *20*
        - Values:
            - Any valid integer
            - *0* (no limit)
    - *page*
        - Type: *integer*
        - Default: *1*
        - Values:
            - Any valid integer, *max = total number of hits / limit + 1*

- *group*
  - Type: *string*
  - Default: *none*
  - Note: types are sorted according to the publisher sorting)
  - Values:
    - *type* – Group by Course Type (courses) or Program Type (programs)
    - *degreetype* – Group by Degree Type (programs)
- *status*
  - Type: *string*
  - Default: *active and active-hidden*
  - Values:
    - *active*
    - *active*-hidden
    - *inactive*

5.4.2 Listing

The *Listing* method allows you to retrieve items from the Acalog database without limiting them with a search query. This method is only available for Courses and Programs. For

Hierarchy Items, see the Get Hierarchy List section.

Example: https://apis.acalog.com/v1/search/courses?**method=listing**

5.4.2.1 Arguments

This method takes a single argument.

Options

- Required: *no*
- Name: *options*
- Type: *array*
  - *sort*
    - Type: *string*
    - Default: *alpha*
    - Values:
      - *alpha*
      - *rank*
  - *Limit*
    - Type: *integer*
    - Default: *20*
    - Values:
      - Any valid integer
      - *0* (no limit)

- *page*
  - Type: *integer*
  - Default: *1*
  - Values:
    - Any valid integer
    - *max = total number of hits / limit + 1*

- *group*
  - Type: *string*
  - Default: *none*
    - types are sorted according to the publisher sorting
  - Values:
    - *type* – Group by Course Type (courses) or Program Type (programs)
    - *degreetype* – Group by Degree Type (programs)

- *status*
  - Type: *string*
  - Default: *active and active-hidden*
  - Values:
    - *active*
    - *active-hidden*
    - *inactive*

- *Modify Date*

  These arguments will allow you to filter the courses by the modified date when making an API call to the Listing method.

  - Type: *string*
  - Default: *none*
  - Options Argument Values:
    - *modify_start* – must be in format of “Y-m-d” (ie: 2012-08-03)
    - *modify_end* – must be in format of “Y-m-d” (ie: 2012-08-03)
  - Example: &options*[modify_start]*=2012-08-03

6. Example Usage

The Acalog Web services API is intended for autonomous use within your systems. This ensures no system IDs are hard-coded and your system can automatically update as your catalog is updated and new revisions are published.

6.1 ABOUT THESE EXAMPLES

While these examples are written in PHP, they use the standard DOM interface, which should closely resemble DOM implementations in many languages. The exact URIs used and responses received are documented in these examples. The HTTP requests themselves have been omitted for brevity.

> *Note: All requests are performed using GET, however the Acalog Web services platform can handle GET or POST requests (or a mix of GET and POST). Simply use the same GET key/value pairs as POST data.*

6.2 FINDING THE CURRENT PUBLISHED CATALOG(S)

All catalogs within Acalog may have a type (typically “Undergraduate” or “Graduate”). Each type may have a single current published catalog and unlimited archived catalogs.

Using this information we can determine the current published catalog in each type by eliminating all archived catalogs. The catalog ID can then be used to make subsequent requests for catalog data, such as courses

6.2.1 Example Code

In this example, we define a `getCurrentCatalogID`() function that will return the current — published, non- archived — catalog for a specific type (optional).

```
<?php
   function getCurrentCatalogID($type = 'no type')  {
   // Your API Key
   $key = ACALOG_PUBLIC_API_KEY;


   // The URL for Acalog Web services platform
   $url = ACALOG_API_URI .
   '/v1/content?format=xml&method=getCatalogs&key='
.$key;
```

```
    // Retrieve the contents of the URL, you could also use
fopen/fread/fclose functions
    $xml  = file_get_contents($url);


    // If an unknown error occurred (i.e. unable to connect)
file_get_contents()return false
    if ($xml === false) {


        return false;
    } else {
        // Create an empty DOMDocument object to hold our
        service response
        $dom = new DOMDocument('1.0', 'UTF-8');


        // Load the XML
        $dom->loadXML($xml);


        // Create an XPath Object
        $xpath = new DOMXPath($dom);


        // Register the Catalog namespace
        $xpath->registerNamespace('c',
        'http://acalog.com/catalog/1.0');


        // Check for error
        $status_elements = $xpath->query('//c:status[text() !=
        "success"]');
        if ($status_elements->length > 0) {
            // An error occurred
            return false;
        }
```

First, we make an HTTP GET request using PHP's built in `file_get_contents()`, available since PHP 4.3.0.

Then, after checking we have a valid response (not false) we load it into a `DOMDocument` using `DomDocument->loadXML()`

Next, we create a `DOMXPath` instance, passing in our `DOMDocument` object. Once we have our `DOMXPath` object, we must register the Acalog `catalog` namespace.

Once we have our document setup, we can then check using XPath to see if an error occurred by checking if the `<c:status>` element contains a value other than success. If an error occurred, the function returns `false`.

```
// Retrieve all catalogs elements
$catalog_elements = $xpath->query('//c:catalog');
if ($catalog_elements->length == 0) {
    // No catalogs found
    return false;
}
```

Next, we use the *DOMXPath* to retrieve all *<c:catalog>* elements within the document. If no elements are found, the function returns false.

```
$catalog_ids = array();

// Iterate over the catalog elements
for ($i = 0; $i < $catalog_elements->length;  $i++) {
    // Get the catalog element
    $catalog_element = $catalog_elements->item($i);
```

Once we have our catalog elements, we iterate over them using a *for* loop, creating an array keyed on the catalog types, and each key containing an array of catalog IDs that belong to that type. This looks like this:

```
array (
    'undergraduate' => array (1,
    3),
    'graduate'  => array (2, 4)
)
```

```
    // Find potential <state><archived /></state>
    elements within the catalog
    $archived_element = $xpath-
    >query('./c:state/c:archived',$catalog_element);

    // If the catalog is archived, move to the next
    iteration
    if ($archived_element->length > 0) {
        continue;
    }
```

We check, using a relative XPath if the *<c:catalog>* element contains a *<c:state><c:archived /></c:state>* element. If so, we move on to the next iteration.

```
    // Get the type
    $type_element = $xpath->query('./c:type',
    $catalog_element);
    if ($type_element->length == 1) {
    $type_name = $type_element->item(0)->nodeValue;
    $type_name = strtolower($type_name);
    } else {
        // Types are not required
        $type_name = 'no type';
    }
```

We get the type for a specific catalog by using a relative XPath for the *<c:type>*, within the context of the *<c:catalog>* element. If no type is found, the type is set to "no type".

```
        // Get the ID attribute
        $id  = $catalog_element->getAttribute('id');


        // Remove the 'acalog-catalog-' from the ID to return
        just the number
        $id = str_replace('acalog-catalog-', '', $id);


        // Add the catalog's ID to the array, indexed  by
        type.
        $catalog_ids[$type_name] = $id;

     }

   }

   // If there is an ID for the type, return it
   if (isset($catalog_ids[$type])) {

   return $catalog_ids[$type];

   } else {

      // No active catalog in the specified type
      return false;

   }

}

// Get the current undergraduate catalog

$current_id = getCurrentCatalogID('undergraduate');

   ?>
```

We get the catalog ID, by retrieving the ID attribute of the *<c:catalog>* element using *DOMElement->getAttribute()*.

Finally, if there is a catalog ID for the specific type, we return the ID. Otherwise, we return false

6.3 RETRIEVE COURSES FOR A SPECIFIC DEPARTMENT

A common use for the Acalog Web services platform is to retrieve course information that updates automatically for display on the departmental pages of your website.

This can be achieved by submitting a course search query with the department name, in the format: parent: ”Department of Art”. Combined with the code from the example above, you can dynamically update the list as new catalogs are published.

6.3.1 Example Code

```
<?php

function getDepartmentCourses($department, $catalog_id) {

    // Your API Key

    $key = ACALOG_PUBLIC_API_KEY;

    // The URL for the Acalog Web services platform

    // Make sure to not limit the number of courses, and sort by
    alpha

    $url = ACALOG_API_URI .
"/v1/search/courses?format=xml&catalog=' .$catalog_id.
'&method=search&key=' .$key.
'&options[sort]=alpha&options[limit]=0&query=';


    // Urlencode the search query

    $query = urlencode('parent:"' .$department. '"');


    $url .= $query;

    // Retrieve the contents of the URL, you could also use
    fopen/fread/fclose functions

    $xml = file_get_contents($url);


    // If an unknown error occurred (i.e. unable to connect)
    file_get_contents() return false

    if ($xml === false) {

        return false;

    } else {

        // Create an empty DOMDocument object to hold our service
        response

        $dom = new DOMDocument('1.0', 'UTF-8');

        // Load the XML

        $dom->loadXML($xml);


        // Create an XPath Object

        $xpath = new DOMXPath($dom);
```

In this example, we define a*getDepartmentCourses*() function that will return the courses assigned to a given department within a specific catalog.

First we define the URL where we will retrieve the data, ensuring that the query value is *urlencoded* using PHP's *urlencode()* function. Note that we set the limit option to zero. This means all courses will be returned.

Then, we make an HTTP GET request using PHP's built in *file_get_contents()*.

Then, after checking we have a valid response (not false) we load it into a *DOMDocument* using *DomDocument->loadXML()*.

Next, we create a *DOMXPath* instance, passing in our *DOMDocument* object.

```
// Check for error
$status_element = $xpath->query('//status[text() !=
"success"]');
if ($status_elements->length > 0) {
    // An error occurred
    return false;
}
```

Once we have our document setup, we can then check using XPath to see if an error occurred by checking if the `<status>` element contains a value other than success. If an error occurred, the function returns `false`.

```
// Check to see if we got any results
$hits_elements = $xpath->query('//hits[text() != "0"]');
if ($hits_elements->length == 0) {
    // No results found
    return false;
}
```

Next, again using XPath, we check the `<hits>` element to see if any results were returned. If no results were found, the function returns false.

```
// Get the result elements
$result_elements = $xpath->query('//result');
```

If there are results, we retrieve all of them using XPath and iterate over then using a `foreach` loop.

```
// Iterate over each course and return an array keyed by
ID
$courses = array();
foreach ($result_elements as $result) {
    // For courses the <altname> is the full course title
    $altname_element = $xpath->query('./altname',
    $result)->item(0);
    $id_element = $xpath->query('./id', $result)->item(0);
    $name = $altname_element->nodeValue;
    $id = $id_element->nodeValue;
    $courses[$id] = $name;
}
```

Again, using XPath we can retrieve the full course title and ID by doing a relative query for the `<altname>` and `<id>` elements respectively within each result.

Each course name is stored in the $`courses` array, keyed on the ID. This looks like this:

```
array (
    6785 => "BIO 101 –
Introduction to Biology",
    6788 => "BIO 208 – Plant
Ecosystems"
)
```

```
        return $courses;
    }
}
$catalog_id = getCurrentCatalogID('undergraduate');
$courses = getDepartmentCourses('Art Education', $catalog_id);


echo '<ul>';
foreach ($courses as $id => $name)  {
    echo '<li><a
href="http://catalog.example.edu/preview_course_nopop.php?coid='
.$id. '&amp;catoid=' .$catalog_id. '">' .htmlentities($name,
ENT_QUOTES, 'UTF-8', true).
'</a></li>' . PHP_EOL;
}
echo '</ul>';
?>
```

Finally, we return the array of courses.

To retrieve the current courses for the *Art Education* department, we first get the current catalog, using the `getCurrentCatalogID()` function defined in the first example, then we call the `getDepartmentCourses()` function.

Next we start an unordered list, using `<ul>`. Then we `foreach` over the courses and output a list item, `<li>`, containing a link to the course, using its ID and the name as the clickable text.

Finally, we end the list using `</ul>`.

6.3.2 Output

```
<ul>
    <li>
        <a href="http://catalog.example.edu/preview_course_nopop.php?coid=6057&amp;catoid=6">
            AEDU 200&nbsp;-&nbsp;Presentation Skills
        </a>
    </li>
    <li>
        <a href="http://catalog.example.edu/preview_course_nopop.php?coid=6058&amp;catoid=6">
            AEDU 201&nbsp;-&nbsp;Introduction  to  Visual Arts  Education
        </a>
    </li>
    <li>
        <a href="http://catalog.example.edu/preview_course_nopop.php?coid=6059&amp;catoid=6">
            AEDU 499&nbsp;-&nbsp;Internship
        </a>
    </li>
    <li>
        <a href="http://catalog.example.edu/preview_course_nopop.php?coid=6060&amp;catoid=6">
            AEDU 501&nbsp;-&nbsp;Creative and Cognitive Development
        </a>
    </li>
    ...
</ul>
```

7. Appendix A: Filter Options

7.1 FILTERS

7.1.1 All Levels > Departments

7.1.1.1 Institution

- Name: description
    - Include link to the institution description
    - Values:
        - no
        - link
- Name: display-name
    - Use actual name of item in "Show/Hide" and description links instead of generic term
    - Values:
        - no
        - yes

7.1.1.2 College

- Name: description
    - Include link to the school/college description
    - Values:
        - no
        - link
- Name: display-name
    - Use actual name of item in "Show/Hide" and description links instead of generic term
    - Values:
        - no
        - yes

7.1.1.3 Department

- Name: display
    - Whether the type of item should be displayed or not, and how
    - Values:
        - yes
- Name: description
    - Include link to the department description
    - Values:
        - no
        - link
- Name: display-name
    - Use actual name of item in "Show/Hide" and description links instead of generic term
    - Values:
        - no
        - yes

7.1.1.4 Sample XML

```
<options>
<institution description="link" display-name="yes"/>
<college description="link" display-name="yes"/>
<department display="yes" description="link" display-name="yes"/>
</options>
```

7.1.2 All Levels > Programs

7.1.2.1 Institution

- Name: description
    - Include link to the institution description
    - Values:
        - no
        - link
- Name: display-name
    - Use actual name of item in "Show/Hide" and description links instead of generic term
    - Values:
        - no
        - yes

7.1.2.2 College

- Name: description

    - Show the description link for the division
    - Values:
        - no
        - link
- Name: display-name
    - Use actual name of item in "Show/Hide" and description links instead of generic term
    - Values:
        - no
        - yes

7.1.2.3 Department

- Name: display
    - Whether the type of item should be displayed or not, and how
    - Values:
        - yes
- Name: sort
    - How the data is sorted
    - Values:
        - hierarchy (as per publisher)
        - name (alphanumeric)
- Name: description
    - Include link to the department description
    - Values:
        - no
        - link
- Name: display-name
    - Use actual name of item in "Show/Hide" and description links instead of generic term
    - Values:
        - no
        - yes

7.1.2.4 Program

- Name: display
    - Whether the type of item should be displayed or not, and how
    - Values:
        - show_hide
        - link

- Name: toggle
  - Display an entire list of programs in a show/hide (Not a specific program)
  - Values:
    - no
    - yes
- Name: group
  - Group programs by this type
  - Values:
    - degree
    - program
    - no
- Name: sort
  - How the data is sorted
  - Values:
    - alpha (alphanumeric)
    - custom (as per publisher custom ordering settings)
- Name: type-descriptions
  - This attribute is optional
  - Include type descriptions
- Values:
  - no
  - yes

7.1.2.5 Sample XML

```
<options>

   <institution description="no" display-name="yes"/>

   <college description="no" display-name="yes"/>

   <department display="yes" sort="hierarchy" description="no" display-name="yes"/>

   <program display="link" toggle="yes" group="program" sort="alpha" type-
   descriptions="yes"/>

</options>
```

7.1.3 Schools/Colleges

7.1.3.1 College

- Name: description
  - Show the description link for the division
  - Values:
    - no
    - link

- Name: display-name
    - Use actual name of item in "Show/Hide" and description links instead of generic term
    - Values:
        - no
        - yes

7.1.3.2 Sample XML

```
<options>
    <college description="link" display-name="yes"/>
</options>
```

7.1.4 Institution > Programs

7.1.4.1 Institution

- Name: description
    - Include link to the institution description
    - Values:
        - no
        - link
- Name: display-name
    - Use actual name of item in "Show/Hide" and description links instead of generic term
    - Values:
        - no
        - yes

7.1.4.2 Program

- Name: display
    - Whether the type of item should be displayed or not, and how
    - Values:
        - link
        - show_hide
- Name: group
    - Group programs by this type
    - Values:
        - degree
        - program
        - no
- Name: sort
    - How the data is sorted
    - Values:
        - alpha (alphanumeric)

    - custom (as per publisher custom ordering settings)
- Name: type-descriptions
  - This attribute is optional o Include type descriptions o Values:
    - no
    - yes

7.1.4.3 Sample XML

```
<options>
    <institution description="link" display-name="yes"/>
    <program display="link" group="program" sort="alpha" type-descriptions="yes"/>
</options>
```

7.1.5 Departments

7.1.5.1 Department

- Name: display
  - Whether the type of item should be displayed or not, and how
  - Values:
    - yes
- Name: sort
  - How the data is sorted
  - Values:
    - name (alphanumeric)
- Name: description
  - Include link to the department description
  - Values:
    - no
    - link
- Name: display-name
  - Use actual name of item in "Show/Hide" and description links instead of generic term
  - Values:
    - no
    - yes

7.1.5.2 Sample XML

```
<options>
    <department display="yes" sort="name" description="link" display-name="yes"/>
</options>
```

7.1.6 Schools/Colleges > Programs

7.1.6.1 College

- Name: description
  - Show the description link for the division
  - Values:
    - no
    - link
- Name: display-name
  - Use actual name of item in "Show/Hide" and description links instead of generic term
  - Values:
    - no
    - yes

7.1.6.2 Programs

- Name: display
  - Whether the type of item should be displayed or not, and how
  - Values:
    - show_hide
    - link
- Name: group
  - Group programs by this type
  - Values:
    - degree
    - program
    - no
- Name: sort
  - How the data is sorted
  - Values:
    - alpha (alphanumeric)
    - custom (as per publisher custom ordering settings)
- Name: type-descriptions
  - This attribute is optional
  - Include type descriptions
  - Values:
    - no
    - yes

7.1.6.2.1 Sample XML

```
<options>
    <college description="link" display-name="yes"/>
    <program display="link" group="degree" sort="custom" type-descriptions="yes"/>
</options>
```

7.1.7 Programs

7.1.7.1 Program

- Name: display
    - Whether the type of item should be displayed or not, and how
    - Values:
        - link
- Name: group
    - Group programs by this type
    - Values:
        - degree
        - program
        - no
- Name: sort
    - How the data is sorted
    - Values:
        - alpha (alphanumeric)
        - custom (as per publisher custom ordering settings)
- Name: type-descriptions
    - This attribute is optional o Include type descriptions o Values:
        - no
        - yes

7.1.7.2 Sample XML

```
<options>
    <program display="link" group="degree" sort="alpha" type-descriptions="yes"/>
</options>
```

7.1.8 Departments > Programs

7.1.8.1 Department

- Name: display
    - Whether the type of item should be displayed or not, and how
    - Values

    - yes
- Name: sort
  - How the data is sorted
  - Values
    - custom
    - hierarchy (as per publisher)
    - name (alphanumeric)
- Name: description
  - Include link to the department description
  - Values
    - no
    - link
- Name: display-name
  - Use actual name of item in "Show/Hide" and description links instead of generic term
  - Values
    - no
    - yes

7.1.8.2 Programs

- Name: display
  - Whether the type of item should be displayed or not, and how
  - Values:
    - show_hide
    - link
- Name: group
  - Group programs by this type
  - Values:
    - degree
    - program
    - no
- Name: sort
  - How the data is sorted
  - Values:
    - alpha (alphanumeric)
    - custom (as per publisher custom ordering settings)
- Name: type-descriptions
  - This attribute is optional o Include type descriptions o Values
    - no
    - yes

7.1.8.3 Sample XML

```
<options>
    <department display="yes" sort="hierarchy" description="link" display-name="yes"/>
    <program display="link" group="program" sort="alpha" type-descriptions="yes"/>
</options>
```

7.1.9 Courses

7.1.9.1 Courses

- Name: display
    - Whether the type of item should be displayed or not, and how
    - Values
        - show_hide
- Name: group
    - Group courses by course type
    - Values
        - no
        - yes
- Name: type-descriptions
    - This attribute is optional o Include type descriptions o Values
        - no
        - yes
- Name: program-display-field
    - Include program display field after each course
    - Values
        - no
        - yes
- Name: sort
    - Custom ordering is not available for this filter
    - How the data is sorted
    - Values
        - no
- Name: prefix
    - Values
        - no
        - yes
- Name: code
    - Values
        - no
        - yes

- Name: name
  - Values
    - no
    - yes

7.1.9.2 Sample XML

```
<options>
    <course display="show_hide" group="yes" type-descriptions="no" program-display-
    field="no">
        <sort custom="no">
            <prefix>no</prefix>
            <code>no</code>
            <name>yes</name>
        </sort>
    </course>
</options>
```

7.1.10 All Levels > Courses

7.1.10.1 Institution

- Name: description
  - Include link to the institution description
  - Values:
    - no
    - link
- Name: display-name
  - Use actual name of item in "Show/Hide" and description links instead of generic term
  - Values:
    - no
    - yes

7.1.10.2 College

- Name: description
  - Show the description link for the division
  - Values:
    - no
    - link
- Name: display-name
  - Use actual name of item in "Show/Hide" and description links instead of generic term

  - Values:
    - no
    - yes

7.1.10.3 Department

- Name: display
  - Whether the type of item should be displayed or not, and how
  - Values
    - yes
- Name: sort
  - How the data is sorted
  - Values:
    - hierarchy (as per publisher)
    - name (alphanumeric)
- Name: description
  - Include link to the department description
  - Values:
    - no
    - link
- Name: display-name
  - Use actual name of item in "Show/Hide" and description links instead of generic term
  - Values:
    - no
    - yes

7.1.10.4 Courses

- Name: display
  - Whether the type of item should be displayed or not, and how
  - Values:
    - show_hide
    - Name: toggle
  - Display an entire list of courses in a show/hide (Not a specific courses)
  - Values:
    - no
    - yes
- Name: group
  - Group courses by course type
  - Values:
    - no
    - yes

- Name: type-descriptions
  - This attribute is optional o Include type descriptions o Values:
    - no
    - yes
- Name: program-display-field
  - Include program display field after each course
  - Values:
    - no
    - yes
- Name: sort
  - Custom ordering is not available for this filter
  - How the data is sorted
  - Values:
    - no
- Name: prefix
  - Values
    - no
    - yes
- Name: code
  - Values:
    - no
  - yes
- Name: name
  - Values:
    - no
    - yes

7.1.10.5 Sample XML

```
<optihons>
    <institution description="link" display-name="yes"/>
    <college description="link" display-name="yes"/>
    <department display="yes" sort="hierarchy" description="link" display-name="yes"/>
    <course display="show_hide" toggle="yes" group="yes" type-descriptions="yes" program-
    display-field="yes">
        <sort custom="no">
            <prefix>yes</prefix>
            <code>yes</code>
            <name>yes</name>
        </sort>
    </course>
</options>
```

7.1.11 Institution > Courses

7.1.11.1Institution

- Name: description
  - Include link to the institution description
  - Values:
    - no
    - link
- Name: display-name
  - Use actual name of item in "Show/Hide" and description links instead of generic term
  - Values:
    - no
    - yes

7.1.11.2 Courses

- Name: display
  - Whether the type of item should be displayed or not, and how
  - Values:
    - show_hide
    - Name: group
  - Group courses by course type
  - Values:
    - no
    - yes
- Name: type-descriptions
  - This attribute is optional
  - Include type descriptions
  - Values:
    - no
    - yes
- Name: program-display-field
  - Include program display field after each course
  - Values:
    - no
    - yes

- Name: sort
    - Custom ordering is not available for this filter
    - How the data is sorted
    - Values:
        - no
- Name: prefix
    - Values:
        - no
        - yes
- Name: code
    - Values:
        - no
        - yes
- Name: name
    - Values:
        - no
        - yes

7.1.11.3 Sample XML

```
<options>
    <institution description="link" display-name="yes"/>
    <course display="show_hide" group="yes" type-descriptions="yes" program-display-
    field="yes">
        <sort custom="no">
            <prefix>yes</prefix>
            <code>yes</code>
            <name>yes</name>
        </sort>
    </course>
</options>
```

7.1.12 Schools/Colleges > Courses

7.1.12.1 College

- Name: description
    - Show the description link for the division
    - Values:
        - no
        - link

- Name: display-name
  - Use actual name of item in "Show/Hide" and description links instead of generic term
  - Values:
    - no
    - Yes

7.1.12.2 Courses

- Name: display
  - Whether the type of item should be displayed or not, and how
  - Values:
    - show_hide
- Name: group
  - Group courses by course type
  - Values:
    - no
    - yes
- Name: type-descriptions
  - This attribute is optional o Include type descriptions o Values:
    - no
    - yes
- Name: program-display-field
  - Include program display field after each course
  - Values:
    - no
    - yes
- Name: sort
  - Custom ordering is not available for this filter
  - How the data is sorted
  - Values:
    - no
- Name: prefix
  - Values
    - no
    - yes
- Name: code
  - Values:
    - no
    - yes

- Name: name
    - Values:
        - no
        - yes

7.1.12.3 Sample XML

```
</options>
    <college description="link" display-name="yes"/>
    <course display="show_hide" group="yes" type-descriptions="yes" program-display-
    field="yes">
        <sort custom="no">
            <prefix>yes</prefix>
            <code>yes</code>
            <name>yes</name>
        </sort>
    </course>
</options>
```

7.1.13 Departments > Courses

7.1.13.1 Department

- Name: display
    - Whether the type of item should be displayed or not, and how
    - Values:
        - yes
- Name: sort
    - How the data is sorted
    - Values:
        - custom (if department has custom sorting, then courses will not have custom sorting)
        - hierarchy (as per publisher)
        - name
- Name: description
    - Include link to the department description
    - Values:
        - no
        - yes
- Name: display-name
    - Use actual name of item in "Show/Hide" and description links instead of generic term

- Values:
  - no
  - yes

7.1.13.2 Courses

- Name: display
  - Whether the type of item should be displayed or not, and how
  - Values:
    - show_hide
- Name: toggle
  - Display an entire list of courses in a show/hide (Not a specific courses)
  - Values
    - no
    - yes
- Name: group
  - Group courses by course type
  - Values:
    - no
    - yes
- Name: type-descriptions
  - This attribute is optional
  - Include type descriptions
  - Values:
    - no
    - yes
- Name: program-display-field
  - Include program display field after each course
  - Values:
    - no
    - yes
- Name: sort
  - How the data is sorted
  - Values:
    - no
    - yes
- Name: prefix
  - Values:
    - no
    - yes

- Name: code
    - Values:
        - no
        - yes
- Name: name
    - Values:
        - no
        - yes

7.1.13.3 Sample XML

```
<options>
    <department display="yes" sort="hierarchy" description="yes" display-name="yes"/>
    <course display="show_hide" toggle="yes" group="yes" type-descriptions="yes" program-
    display-field="yes">
        <sort custom="no">
            <prefix>yes</prefix>
            <code>yes</code>
            <name>yes</name>
        </sort>
    </course>
</options>
```

7.1.14 Archived Catalogs

This filter has no template options

7.1.15 School/Colleges

7.1.15.1 College

- Name: description
    - Show the description link for the division
    - Values:
        - no
        - link
        - inline
- Name: display-name
    - Use actual name of item in "Show/Hide" and description links instead of generic term
    - Values:
        - no
        - yes

7.1.15.2 Department

- Name: display
    - Whether the type of item should be displayed or not, and how
    - Values:
        - yes
- Name: sort
    - How the data is sorted
    - Values:
        - hierarchy (as per publisher)
        - name (alphanumeric)
- Name: description
    - Include link to the department description
    - Values:
        - no
        - link
        - inline
- Name: display-name
    - Use actual name of item in "Show/Hide" and description links instead of generic term
    - Values:
        - no
        - yes

7.1.15.3 Programs

- Name: display
    - Whether the type of item should be displayed or not, and how
    - Values:
        - show_hide
        - link
        - no
- Name: toggle
    - This attribute is optional
    - Display an entire list of programs in a show/hide (Not a specific program)
    - Values:
        - no
        - yes
- Name: group
    - This attribute is optional
    - Group programs by this type
    - Values:

        - degree
        - program
        - no
- Name: type-descriptions
    - This attribute is optional o Include type descriptions o Values:
        - no
        - yes
- Name: sort
    - This attribute is optional
    - How the data is sorted
    - Values:
        - alpha (alphanumeric)
        - custom (as per publisher custom ordering settings)

7.1.15.4 Courses

- Name: display
    - Whether the type of item should be displayed or not, and how
    - Values:
        - details
        - link
        - no
- Name: toggle
    - This attribute is optional
    - Display an entire list of courses in a show/hide (Not a specific course)
    - Values:
        - no
        - yes
- Name: group
    - This attribute is optional
    - Group courses by course type
    - Values:
        - no
        - yes
- Name: type-descriptions
    - This attribute is optional o Include type descriptions o Values:
        - no
        - yes
- Name: program-display-field
    - This attribute is optional
    - Include program display field after each course
    - Values:

        - no
        - yes
- Name: sort
    - How the data is sorted
- Name: custom
    - Values:
        - no
        - yes
- Name: prefix
    - Values:
        - no
        - yes
- Name: code
    - Values:
        - no
        - yes
- Name: name
    - Values:
        - no
        - yes

7.1.15.5 Sample XML

```
<options>
    <college description="inline" display-name="yes"/>
    <department display="yes" sort="hierarchy" description="inline" display-name="yes"/>
    <program display="link" toggle="yes" group="degree" type-descriptions="yes"
    sort="alpha"/>
    <course display="details" toggle="yes" group="yes" type-descriptions="yes" program-
    display-field="yes">
        <sort custom="no">
            <prefix>yes</prefix>
            <code>yes</code>
            <name>yes</name>
        </sort>
    </course>
</options>
```

7.1.16 Department

7.1.16.1 Department

- Name: display
    - Whether the type of item should be displayed or not, and how
    - Values:
        - yes
- Name: sort
    - How the data is sorted
    - Values:
        - hierarchy (as per publisher)
        - name (alphanumeric)
- Name: description
    - Include link to the department description
    - Values:
        - no
        - link
        - inline
- Name: display-name
    - Use actual name of item in "Show/Hide" and description links instead of generic term
    - Values:
        - no
        - yes

7.1.16.2 Programs

- Name: display
    - Whether the type of item should be displayed or not, and how
    - Values:
        - show_hide
        - link
        - no
- Name: group
    - This attribute is optional
    - Group programs by this type
    - Values:
        - degree
        - program
        - no
- Name: sort
    - This attribute is optional
    - How the data is sorted

  - Values:
    - alpha (alphanumeric)
    - custom (as per publisher custom ordering settings)
- Name: type-descriptions
  - This attribute is optional o Include type descriptions o Values:
    - no
    - yes

7.1.16.3 Courses

- Name: display
  - Whether the type of item should be displayed or not, and how
  - Values:
    - details
    - link
    - no
- Name: toggle
  - This attribute is optional
  - Display an entire list of courses in a show/hide (Not a specific course)
  - Values:
    - no
    - yes
- Name: group
  - This attribute is optional
  - Group courses by this value
  - Values:
    - no
    - yes
- Name: type-descriptions
  - This attribute is optional
  - Include type descriptions
  - Values:
    - no
    - yes
- Name: program-display-field
  - This attribute is optional
  - Include program display field after each course
  - Values:
    - no
    - yes
- Name: sort
  - How the data is sorted

- Values:
  - no
  - yes
- Name: prefix
  - Values:
    - no
    - yes
- Name: code
  - Values:
    - no
    - yes
- Name: name
  - Values:
    - no
    - yes

7.1.16.4 Sample XML

```
<options>
    <department display="yes" sort="hierarchy" description="inline" display-name="yes"/>
    <program display="link" group="degree" sort="alpha" type-descriptions="yes"/>
    <course display="details" toggle="no" group="yes" type-descriptions="yes" program-
    display-field="yes">
        <sort custom="no">
            <prefix>yes</prefix>
            <code>yes</code>
            <name>yes</name>
        </sort>
    </course>
</options>
```

7.1.17 Program

This filter has no template options

7.1.18 Catalog Search

7.1.18.1 Search

- Name: query
  - Value: The query that you are searching for

- Name: course
    - Values:
        - no
        - yes
- Name: program
    - Values:
        - no
        - yes
- Name: entity
    - Values
        - no
        - yes
- Name: other
    - Values
        - no
        - yes

7.1.18.2 Sample XML

```
 <options>
   <search>
      <query>"Math"</query>
      <course>yes</course>
      <program>yes</program>
      <entity>yes</entity>
      <other>no</other>
   </search>
</options>
```

8. Appendix B: Implemented Standards and Specifications

Acalog Web services platform uses industry standard XML as determined by the W3C (World Wide Web Consortium) and implements several difference specifications in addition to Acalog custom XML.

- XHTML 1.0 [http://www.w3.org/TR/xhtml1/]
- XML Inclusions 1.0 (XInclude) [http://www.w3.org/TR/xinclude/]
- XML Pointer Language 1.0 (XPointer) [http://www.w3.org/TR/WD-xptr]
- The Atom Syndication Format (RFC 4287) [http://www.ietf.org/rfc/rfc4287.txt