Avalon Ingest API

A lightweight API for indexing metadata and links to existing derivatives into Avalon.

Version	Date	Description
0.1	13 October 2015	Initial Draft
0.2	16 December 2015	API as implemented

Table of Contents

access control

All API methods are protected by token authentication. A specified token is passed through http header 'Avalon-Api-Key'. A matching token must be configured in in Avalon's database. Creating and viewing tokens can be done via rake tasks.

mediaobjects

GET media_objects/:pid.json Retrieves a subset of the mods for the specified media object

GET media_objects.json Retrieves a paged list of media_objects (parameters :page and :per_page specify page number and number per page for pagination)

POST media_objects.json Creates an media object

PUTS media_objects/:pid.json Updates an media object

parameters for POST and PUTS:

:fields – mods converted to JSON hash. required fields: title, date_issued

:title (required),
:creator (multiple),
:date_issued (required),
:alternative_title (multiple),
:translated_title (multiple),
:uniform_title (multiple),
:statement_of_responsibility,
:date_created,
:copyright_date,
:abstract,
:note (multiple, requires :note_type from controlled vocabulary to be present also)
:format,
:resource_type (multiple),
:contributor (multiple),
:publisher (multiple),
:genre (multiple),
:subject (multiple),
:related_item_url (multiple, requires :related_item_label to be present also),
:geographic_subject (multiple),
:temporal_subject (multiple),
:topical_subject (multiple),
:bibliographic_id,
:language (multiple),
:terms_of_use,
:table_of_contents (multiple),
:physical_description,
:other_identifier (multiple, requires :other_identifier_type from controlled vocabulary to be present also. In the MDPI case use "mdpi barcode"),
:comment (multiple)

:collection_id

:files – an array masterfile hashes, example below. label, structure, and captions are optional. The last five fields must have the specified values (for now).

:files example

```
       [{label: "Part 1",
                  title: "Part 1",
        files: [{label: 'quality-high',
                  id: 'track-1',
                  url: "https://streaming.server/path/to/high.m3u8",
                  duration: "6315",
                  mime_type:  "video/mp4",
                  audio_bitrate: "127716.0",
                  audio_codec: "AAC",
                  video_bitrate: "1000000.0",
                  video_codec: "AVC",
                  width: "640",
                  height: "480" },
                 {label: 'quality-medium',
                  id: 'track-2',
                  url: "https://streaming.server/path/to/medium.m3u8",
                  duration: "6315",
                  mime_type: "video/mp4",
                  audio_bitrate: "127716.0",
                  audio_codec: "AAC",
                  video_bitrate: "1000000.0",
                  video_codec: "AVC",
                  width: "640",
                  height: "480" }
                ],
        file_location: "/srv/avalon/master_files/file.mp4",
        file_checksum: "7ae24368ccb7a6c6422a14ff73f33c9a",
        file_size: "199160",
        duration: "6315",
        display_aspect_ratio: "1.7777777777777777",
        original_frame_size: "640x480",
        file_format: "Moving image",
        poster_offset: "0:02",
        thumbnail_offset: "0:02",
        date_digitized: "2015-12-31",
        structure: structure xml,
        captions: captions text from vtt or srt file,
        captions_type: 'text/vtt' (or 'text/srt')
            other_identifier: ["40000000045312"],
        comment: ["test comment"],
        structure: "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n<!-- Music for Piano; http://server1.
variations2.indiana.edu/variations/cgi-bin/access.pl?id=BFJ6801 -->\n<Item label=\"CD 1\">\n    <Div label=\"
Copland, Three Piano Excerpts from Our Town\">\n        <Span label=\"Track 1. Story of Our Town\" begin=\"0\"
end=\"0:09.99\"/>\n        <Span label=\"Track 2. Conversation at the Soda Fountain\" begin=\"0:10\" end=\"0:
19.99\"/>\n        <Span label=\"Track 3. The Resting Place on the Hill\" begin=\"0:20\" end=\"0:29.99\"/>\n
</Div>\n    <Div label=\"Copland, Four Episodes from Rodeo\">\n        <Span label=\"Track 4. Buckaroo
Holiday\" begin=\"0:30\" end=\"0:39.99\"/>\n        <Span label=\"Track 5. Corral Nocturne\" begin=\"0:40\"
end=\"0:49.99\"/>\n        <Span label=\"Track 6. Saturday Night Waltz\" begin=\"0:50\" end=\"0:59.99\"
/>\n        <Span label=\"Track 7. Hoe-Down\" begin=\"1:00\" end=\"1:09.99\"/>\n    </Div>\n    <Span label=\"
Track 8. Copland, Piano Variations \" begin=\"1:10\" end=\"1:19.99\"/>\n    <Div label=\"Copland, Four Piano
Blues\">\n        <Span label=\"Track 9. For Leo Smit: Freely poetic\" begin=\"1:20\" end=\"1:29.99\"
/>\n        <Span label=\"Track 10. For Andor Foldes: Soft and languid\" begin=\"1:30\" end=\"1:39.99\"
/>\n        <Span label=\"Track 11. For Willian Kapell: Muted and sensuous\" begin=\"1:40\" end=\"1:49.99\"
/>\n        <Span label=\"Track 12. For John Kirkpatrick: WIth bounce\" begin=\"1:50\" end=\"1:59.99\"/>\n    <
/Div>\n    <Span label=\"Track 13. Copland, Danzon Cubano\" begin=\"2:00\" end=\"2:30\"/>\n</Item>\n",
        workflow_name: "avalon",
        percent_complete: "100.0",
        percent_succeeded: "100.0",
        percent_failed: "0",
        status_code: "COMPLETED"
}]
```

:import_bib_record – boolean. If true, fields must include value for :bibliographic_id and may include value from controlled vocabulary for :bibliographic_id_label)

Bib import failure will result in a JSON response: `{errors: ['Bib import failed', e.message]}, statu`

:replace_masterfiles - boolean. Relevant only if the media object already exists and it has masterfiles. If true, existing masterfiles will be replaced by those sent. If false, sent masterfiles will be appended to existing list.

:publish – boolean. If true, mediaobject will automatically enter a published state with avalon_publisher='REST API'

admin/collections

GET admin/collections.json Retrieves a paginated list of all collections

GET admin/collections/:pid.json Retrieves information on the collection

GET admin/collections/:pid/items.json Retrieves a paginated list of all items in collection :pid

POST admin/collections.json Creates a collection

parameters for POST and PUT:

admin_collection:

name: collection name,
description: collection description (optional),
unit: collection unit from controlled vocabulary,
managers: array of Avalon user ids or emails

PUTS admin/collections/:pid.json Updates a collection

units

In avalon, units are part of a controlled vocabulary. To add a new unit, a call to the vocabulary is required.

GET vocabulary.json Retrieves a hash of avalon controlled vocabularies, { units: ['Default Unit' , ...], ... }

GET vocabulary/:vocab.json Retrieves values contained in specified controlled vocabulary

POST vocabulary/:vocab.json Updates specified vocab by adding the value contained in the :entry parameter

response codes:

Code	Description
200	Okay. For GET, the relevant JSON will be contained in response body. For PUTS and POST response body will contain pid of created/updated item.
201	<Not used> Created
202	<Not used> Accepted, Queued
400	<Not used> Bad request
401	Auth Failure. No token present in request header 'Avalon-Api-Key'
403	Forbidden. Invalid token in request header 'Avalon-Api-Key'
404	Resource Not Found. If object is not found for requested pid.
405	<Not used> Method Prohibited
409	<Not used> Conflict/Other Error. Conflict ex: pid in use
422	Processing failed. Response body will contain JSON hash with :errors key pointing to a list of error message strings.