# ClipFile Web Service API

version 2.0

last modified 7/22/12

# Overview

This document sketches out the order of development and testing on ClipFile Web Service using version 2.0 of the web service api.

All requests are HTTPS GET unless otherwise pointed out.

# Resource Methods

Each resource method handles a different GET or POST request.

## getServiceApiVersion()

This method returns the version number of the RESTful API of this web service in a.b format where

a = major version number  
b = minor version number

The API version number is included in the header of all result pages.

Note that ClipFile WebService API version 2.0 is not compatible with version 1.0.

## getServiceBuildNumber()

This method returns the build number of this web service in a.b.c.d format where

a = major version number  
b = minor version number  
c = milestone number  
b = source revision number

The service build number is included in the header of all result pages.

## AuthenticationService

This service handles login, session creation, and session refresh using

AuthenticationService.login()

AuthenticationService.validateCredentials()

AuthenticationService.createSession()

AuthenticationService.getSession()

AuthenticationService.removeExpiredSession()

AuthenticationService.logout()

This service has a configurable sessionExpireMin /default 30/ which sets the expiration timeout in minutes for all new sessions. A background thread invokes

AuthenticationService.deleteExpiredSessions()

to find and delete expired sessions every sessionExpireCheckMin /default 1/ minutes.

## /loginUser?email,password

A registered user provides email and password\* at login. These credentials are then used by

AuthenticationService.login()

AuthenticationService.validateCredentials ()

To lookup an existing user account. If a valid user account is found using

AccountService.findAccount()

then a new session is created using

AuthenticationService.createSession()

The ID of this session is required for all other API calls. This sessionID expires after a default of 30 minutes of inactivity but each session is refreshed upon each call of

AuthenticationService.getSession()

When a user attempts to login, the success or failure of this request will be returned using

AuthenticationResultsPage

\*Passwords are encrypted by the client using SHA256 hash coding with salt and are then stored in the server database without change.

## /logoutUser?sessionID

This GET request expires the session of the logged in user using

AuthenticationService.logout()

SessionInfo.expireNow()

AuthenticationService.removeExpiredSession()

Returns a confirmation message using

AuthenticationResultsPage

## /getItems?sessionID,selectExp,itemsPerPage,pg

This GET request uses the select expression and pagination parameters to find and load a page of sorted items with their associated attributes. This search is done using

BaseDao.findPageOfItems()

Returns uri + select expression + a page of sorted items + profiling using

ItemSearchResultsPage

## /getRelatedTags?sessionID,tagText,scope\*,r-search\*\*

This GET request finds all tags related to the given tagText within the given scope and search parameters using

RelatedTagService.findPageOfRelatedTags()

Returns uri + select expression + a page of sorted TagText values + profiling using

RelatedTagSearchResultsPage

Due to the high performance cost of this operation, this query is made against a RelatedTagCache that is periodically recomputed every relatedTagsRecomputeMin /default 60/ minutes. The cache update is done using a background thread that invokes

RelatedTagService.recomputeRelatedTags()

\* "scope" (previously referred to as "filter") is a filtering parameter that has a default value of "ANY" but also has values "NETWORK" or "USER" (previously called "Mine")

\*\* "r-search" means any filtering, sorting and pagination params used to find the most recently computed related tags in the RelatedTagCache.

## /getClips?sessionID,scope,e-search\*

This GET request finds a page of shallow\*\* clips that match the given scope and entity search parameters using

EntityService.findPageOfEntities()

Returns uri + select expression + e-search + page of shallow\* entities + profiling using

EntitySearchResultsPage

\*"e-search" means any filtering, sorting and pagination params used to fetch Entities from the server database.

\*\* "shallow" means that the result includes no highlights or tags other than subject, topic, title, subtitle, source, author, and publishdate

## /getClip?sessionID,clipID

This GET request loads a clip using

EntityService.getEntity()

returns uri + select expression + a fully-loaded\* clip + profiling using

EntityServiceResultsPage

\* "fully-loaded" means that all descendants have been found and each has been loaded with full S3 data. Entities with S3 data include the clip itself, the highlights, and the web page.

## /getAuthors?sessionID,clipID,scope,t-search\*

This GET request finds authors within the given scope and entity search parameters using

TagTextService.findPageOfTagTexts()

Returns uri + select expression + t-search + a page of sorted text elements + profiling using

TagTextSearchResultsPage

\* "t-search" refers to all parameters needed to search for tags, most importantly by TagClass = “author”

## /getSources?sessionID,clipID,scope,t-search

This GET request finds sources within the given scope and entity search parameters using

TagTextService.findPageOfTagTexts()

Returns uri + select expression + t-search + a page of sorted text elements + profiling using

TagTextSearchResultsPage

\* "t-search" refers to all parameters needed to search for tags, most importantly by TagClass = “source”

## /getTags?sessionID,parentID,scope,e-search

This GET request finds tags that match the given scope and e-search using

EntityService.findPageOfEntities()

Returns uri + select expression + e-search + a page of sorted highlights + profiling using

EntitySearchResultsPage

This request is used to get:

* all tags of a given clip
* all tags of a given highlight
* all tags of a given tags
* all tags of the logged-in user
* all tags

## /getHighlights?sessionID[,clipID],scope,e-search

This GET request finds highlights that match the given scope and e-search using

EntityService.findPageOfEntities()

Returns uri + select expression + e-search + a page of sorted highlights + profiling using

EntitySearchResultsPage

This request is used to get:

* all highlights of a given clip
* all highlights of the logged-in user
* all highlights

## /deleteEntity?sessionID,entityID

This GET request deletes the given entity using

EntityService.deactivateEntity()

Returns uri + update expression + updated entity info + profiling using

EntityServiceResultsPage

## /updateEntityForm?sessionID,entityID,e-props\*

This POST request updates the given entity using

EntityService.updateEntity()

Returns uri + update expression + updated entity info + profiling using

EntityServiceResultsPage

\* "e-props" means any editable properties of a given entity.

## /updateEntity?sessionID,entityID,e-props

This GET request takes the given e-props to update a pre-existing entity. Since this is a GET request, the total length of the URI should not exceed a max length of 2,000\* characters. A single entity is updated using

EntityService.updateEntity()

Returns uri + update expression + updated entity info + profiling using

EntityServiceResultsPage

\* Although the specification of the HTTP protocol does not specify any maximum length for URIs practical limits are imposed by web browser and server software. The generally accepted rule is to not let the URI exceed 2,000 characters.

## /addTag?sessionID,parentID,TagText[,TagClass], e-props

This GET request adds a new tag to the given clip, highlight or tag using

EntityService.createTag()\*

Returns uri + create expression + new entity info + profiling using

EntityServiceResultsPage

\*Note that unlike most GET requests this add request is not safe and is not idempotent because multiple GET requests can continue adding new identical tags.

## /addHighlightForm?sessionID,clipID,e-props

This POST request adds a new highlight to the given clip using

EntityService.createHighlight()\*

Returns uri + create expression + new entity info + profiling using

EntityServiceResultsPage

\*This add request is not idempotent.

## /copyClip?sessionID,clipID

This GET request uses data from the referenced clip to make a duplicate copy for the logged-in user using

EntityService.copyEntity()\*

Returns uri + select & create expressions + new clip info + profiling using

EntityServiceResultsPage

\*This copy request is not idempotent.

## /shareClip?sessionID,clipID,from,recipients\*

This GET resource method takes a clip and "shares" it with the given recipients via email. Sharing is done using

EntityService.shareEntity()

Returns uri + sharing info + profiling using

EntityServiceResultsPage

\*”recipients” is a multi-value query parameter that takes accountIDs or email addresses with optional name.

## /getAccount?sessionID

This GET request finds the account of the current user using

AccountService.getAccount()

Returns uri + select expression + account info + profiling using

AccountServiceResultsPage

## /getAccounts?sessionID,scope,a-search\*

This GET request finds all of the accounts and data that the current user is authorized to see using

AccountService.findPageOfAccounts()

Returns uri + select expression + a page of accounts + profiling using

AccountSearchResultsPage

\* "a-search" refers to all parameters needed to find accounts that the current user is authorized to see.

## /addAccountForm?sessionID,a-props

This PUT request creates a new user account using several account properties (a-props) including accountName, firstName, lastName, email, password, and role. The email and role parameters can be multi-valued. This is handled by

AccountService.createAccount()\*

Returns uri + create expression + user info + profiling using

AccountServiceResultsPage

\*This add request is not idempotent.

## /addAccount?sessionID,a-props

This GET request creates a new user account using several account properties (a-props) using

AccountService.createAccount()

Returns uri + select expression + user detail + profiling using

AccountServiceResultsPage

\*This add request is not idempotent and the uri should not exceed 2,000 characters.

## /updateAccountForm?sessionID,accountID,a-props

This PUT request updates an existing user account using several account properties (a-props) including accountName, firstName, lastName, email, password, and role. The email and role parameters can be multi-valued. This update request uses

AccountService.updateAccount()

Returns uri + create expression + user info + profiling using

AccountServiceResultsPage

## /updateAccount?sessionID,accountID,a-props

This GET request updates an existing user account using several account properties (a-props) including accountName, firstName, lastName, email, password, and role. The email and role parameters can be multi-valued. This update request uses

AccountService.updateAccount()

Returns uri + create expression + user info + profiling using

AccountServiceResultsPage

## /deleteAccount?sessionID,accountID

This GET request is used to delect (actually deactivate) an existing user account using

AccountService.deleteAccount()

Returns uri + update expression + a-props + user detail using

AccountServiceResultsPage

## /addClipForm?sessionID[,url],title,source,author,publish date,e-props

Requirements to be determined for this POST request.

**Pages**

## AuthenticationResultsPage(service, uri, errors)

* uses headerToXml(pageName)
* uses AuthenticationService.getSessionInfo()
* uses sessionInfoToXml(info,indent)
* uses errorsToXml(errors,indent)
* uses accountToXml(account,indent)
* uses profilingToXml(indent)
* uses footerToXml(pageName)

1. **EntitySearchResultsPage(service, uri, EntitySearchCmd search, List<Entity>entities, errors)**

* uses headerToXml(pageName)
* uses entitySearchToXml(search,indent)
* uses errorsToXml(errors,indent)
* uses entitiesToXml(entities,indent)
* uses profilingToXml(indent)
* uses footerToXml(pageName)

1. **EntityServiceResultsPage(service,uri, serviceType,Entity entity,errors)**

* uses headerToXml(pageName)
* uses serviceTypeToXml(serviceType,indent)
* uses errorsToXml(errors,indent)
* uses entityToXml(entity,indent)
* uses profilingToXml(indent)
* uses footerToXml(pageName)

1. **ItemSearchResultsPage(service, uri, queryExp,errors)**

* uses headerToXml(pageName)
* uses itemQueryToXml(queryExpression,indent)
* uses errorsToXml(errors,indent)
* uses itemsToXml(items,indent)
* uses profilingToXml(indent)
* uses footerToXml(pageName)

1. **TagTextSearchResultsPage(service, uri, tagTextSearch, tagTexts, errors)**

* uses headerToXml(pageName)
* uses tagSearchToXml(search,indent)
* uses errorsToXml(errors,indent)
* uses tagTextsToXml(tagTexts,indent)
* uses profilingToXml(indent)
* uses footerToXml(pageName)

1. **AccountSearchResultsPage(service, uri, search, accounts, errors)**

* uses headerToXml(pageName)
* uses accountSearchToXml(search,indent)
* uses errorsToXml(errors,indent)
* uses accountsToXml(accounts,indent)
* uses profilingToXml(indent)
* uses footerToXml(pageName)

1. **AccountServiceResultsPage(service, uri, serviceType, account, errors)**

* uses headerToXml(pageName)
* uses serviceTypeToXml(serviceType,indent)
* uses errorsToXml(errors,indent)
* uses accountToXml(account,indent)
* uses profilingToXml(indent)
* uses footerToXml(pageName)

**XML Render Methods**

## WebService.RenderUtil

## accountSearchToXml(search,indent)

## accountToXml(account,indent)

## accountToXml(account,indent)

## accountsToXml(accounts,indent)

## entitiesToXml(entities,indent)

## entitySearchToXml(search,indent)

## entityToXml(entity,indent)

## errorsToXml(errors,indent)

## footerToXml(pageName)

## headerToXml(pageName)

## itemQueryToXml(queryExpression,indent)

## itemsToXml(items,indent)

## profilingToXml(indent)

## sessionInfoToXml(info,indent)

## tagSearchToXml(search,indent)

## tagTextsToXml(tagTexts,indent)

**Service Methods**

## AccountService

## createAccount()

## .deleteAccount()

## findPageOfAccounts()

## getAccount()

## updateAccount()

## AuthenticationService

## authenticateUser()

## createSession()

## deleteExpiredSessions()

## expireSession()

## getSession()

## BaseDao

## findPageOfItems()

## EntityService

## copyEntity()

## createEntity()

## createHighlight()

## createTag()

## deleteEntity()

## findPageOfEntities()

## getEntity()

## shareEntity()

## updateEntity()

## RelatedTagService

## findPageOfRelatedTags()

## TagTextService

## findPageOfTagTexts()