

Generic Agency Contributor API - Design Outline

There are a wide range of concerns when designing an API (Application Programming Interface) capable of being used by a various service providers. The design must be flexible enough to support the presence/absence of various information as differ at each data-source, yet must be solid enough to allow for the reliable operation of the functionality provided by each.

The document outlines first the most basic functions required to support contribution of images and common related data across various image agencies, and secondly the functions required for additional tasks relevant to a minority of potential hosts.

The intent is to specify a system whereby images may be received and stored, and relevant information pertaining to each image can be registered to accompany each image. Whilst maintaining appropriate references of ownership to the image creator, provider or rights-holder.

In addition, the retrieval of a contributor's own data (that which they rightly own) relevant to their portfolio of images, and related sales & performance data shall be retrievable by this design to aid in their business analysis and planning.

Benefits of Providing a Contributor API

Very briefly, some of the benefits of of providing an API including contributor functionality are:

- Contributors workflow & efficiency can be improved significantly, allowing them more time to create (and improve) images or content suitable to your market.
- Contributors will be capable of assessing their statistics in detail, quickly. Effectively outsourcing the 'leg-work' of analysing your data to determine in which fields you are (or are not) performing well, in bite-size chunks to those people most interested in ensuring their own success (along with yours).

There is a more thorough accompanying document entitled: "[Benefits of Providing a Contributor API](#)" which covers in detail the benefits an agency and it's contributors may realise from implementation of this design.

Terminology

- The term 'image' may apply to either a single image file, often in jpeg format (though unrestricted as such), or to an image-pair: jpg+raw or jpg+vector etc.
 - The image content must match for a pair, and thus metadata and ownership relevant to each file of an image-pair applies equally to the image-pair as a whole.
 - An image-pair may not be restricted to a maximum number of files, as in the case of jpg+svg+tiff, though support for such restrictions should be in place.
- The term 'content' applies as does image, but may also refer to non-image creative content. This may include video, audio, flash, or even written works.
 - It is not limited to the above, but it is assumed that the work is digital in nature, or has been digitised for storage/distribution.
 - This design primarily refers to 'image' but within it's scope, the term 'content' may be considered synonymous.
- The term 'metadata' applies generally to any information used to describe any element of content, or aspect thereof. This includes it's owner and ownership data.
- The term 'owner' refers to the content originator, creator, or rights-holder of the associated content.
 - This 'owner' is termed as such both in reference to their copyright ownership (or legal control thereof), and their right under many data-protection laws for their legal right to access 'within reasonable means' any information or data pertaining to themselves or their assets.
- The term 'third party' refers to any API user who is not accessing their own (and owned) data, but is instead performing operations at the request of a data-owner, or at the granted permission or request of the API host.

Assumptions & Considerations

It is assumed that the following requirements are to be met:

- The design is intended as an access-route to existing systems: maintenance and sales operations should be performed by existing systems. This design is intended only to provide services for contributors. The remainder of the image-sale agency business model is outside the scope of this design.
- Implementation of the various functions will differ as per the operating model of each potential API host. The design is to outline a standard system of access (interface specification), but not to determine exact implementation methods.
 - Implementation will vary based on host infrastructure, though it is assumed that hosts store their data in databases accessible using SQL.
- For image uploads, the API need not handle file-management, it is assumed existing infrastructure is in place (which need not be replicated). Existing code may be used by providing a method on the API providing the upload destination and default form fields, or FTP information as per the requirements of each host system.
- The different structure of hosts data stores require that most field values are to be considered optional. Requiredness of data shall be specifiable at the implementation level.
 - In addition, the order in which some functions must be executed may differ: ie. for some hosts it may be required to upload, then provide metadata, for others it may require metadata, then upload.
- Account administration shall remain outside the scope of this design.
- Image-buyer API functions, although outside of the scope of this design document, should be a consideration of the broader design of the platform.
 - The contributors design could be expanded to image buyers easily with the addition of functions such as: `get_SearchResults`, `get_PurchasedImageList`, and `get_Lightbox`.

Primary Functions

The primary functions are those relevant to only the majority of systems.

Function	Parameters	Description
Image Management:		
put_ImageMetadata	Title, Description, Keywords, Language, Content Type, Ref No, Filename, Owner Id	Creates (or updates) an image record with provided metadata, with or without references to identify the file(s) or uploaded content.
get_UploadDestination	Ref No, Content Type, Owner Id	Returns a set of data specifying the upload location.
put_UploadImage	File(s), Content Type, Ref No, Owner Id	Receives an uploaded file (with or without a reference) and stores in or sends to the appropriate systems.
get_ImageMetadata	Ref No	Returns the metadata associated with the provided image reference.
get_ImageList	Owner Id, Phrase, Options	Returns a list of image references for that owner (optionally matching provided search phrase or other options)
Release Management:		
put_Release	Release Type, Name, DoB, Location, File, Owner Id	Receives an uploaded file and stores in or sends to the appropriate systems.
get_Releases	Release Type, Ref No, Options, Owner Id	Returns a list of releases for this owner (optionally matching search options)
assign_Releases	Release(s), Ref No, Owner Id, Remove	Assigns provided release records to provided image reference.
Authentication:		
Login	Username, Encrypted Password	Creates a session for this user (if stateful), or returns a value confirming they are sufficiently authenticated to alter information relating to their account (which may be provided to the api for future operations).
Logout	Owner Id	Closes a session for this user or any user if omitted.

Prevents further
authenticated operations.

Workflow & Status:

get_ImageStatus	Ref No, Owner Id	Returns review status for image reference (in review, rejected (and reason), approved)
get_ImageHistory	Ref No, Owner Id, Start Date, End Date	Returns sales records for an image record
get_AccountHistory	Owner Id, Start Date, End Date	Returns sales records across all images for a given owner account
get_AccountBalance	Owner Id	Returns the current account total earnings
get_PortfolioCount	Owner Id	Returns the total count of images available for sale in portfolio

Secondary Functions

The secondary functions are those likely relevant to a minority of systems.

Function	Arguments	Description
Satellite Metadata:		
assign_Licenses	Ref No, Licenses, Owner Id	Sets the granted licenses on referenced image to provided licenses.
assign_Categories	Ref No, Categories, Owner Id	Sets the provided categories on referenced image.
assign_Vocabulary	Ref No, Vocabulary Set, Owner Id	Sets the provided vocabulary on referenced image.
Lookups:		
get_ContentTypes		Returns a list of content types (image, video, audio, etc)
get_Licenses	Content Type	Returns a list of licenses for this content type (each named value should include a link to relevant legal documentation).
get_Categories	Content Type, Keyword(s)	Returns a list of categories for this content type (including Ids and/or matching provided keywords).
get_Vocabulary	Keyword(s)	Returns a list of vocabulary matches (matching against provided keyword(s)) for systems with a fixed vocabulary restriction.

Implementation Guidelines

When considering the distribution of data from any source, there are many issues to consider outside of the technical requirements.

Although the following issues are not necessarily aspects controllable within the scope of the outline design for the API framework, they are crucial to the successful (and legal) operation of an API with the intent of data distribution.

These issues should be considered as key aspects of development for any host implementing their API on the aforementioned design:

- Privacy laws dictate that a contributor (or other API user) shall not receive data usable to identify any other contributor (or API user). This means sales/performance data must be anonymised.
 - In addition, whereby a third-party is explicitly granted access to the API on behalf of a contributor, by said contributor: the data-sets available to be returned shall not include data capable of personally identifying the contributor beyond that which they have already provided to the third-party.
 - In any scenario which may allow third-parties to access data (general performance or sales data for example), any data pertaining to the data-owner must be removed completely or sufficiently anonymised to prevent privacy breaches.
- OAuth is omitted from this design as an implementation of the security model due to it's requirement for specific database storage requirements, however any 'Owner Id' parameter below could be substituted for it's OAuth-token equivalent.
- Different hosts, in different geographical locations, will each face different legal requirements. The design specifies the bare minimum of required functionality to allow for minimal potential legal impact.
 - Hosts wishing to build beyond the basically specified functionality may of course do so, and are encouraged to do so for data they can legally make available (that which is non-user-owned or can be anonymised sufficiently).
- If implementing the API as stateless (no session allowed) the Owner Id shall be determined at the point of call by verifying provided credentials (username and encrypted password).
 - If stateful, the Owner Id provided by the caller must be verified against the Owner Id stored in the session (ie. a login call must precede or accompany any request requiring authentication).
 - If the host chooses to implement OAuth this may act as the Owner Id (dependent upon how tightly the host chooses to integrate OAuth with the data source).

This outline is intended to be built upon the apiKit framework. I have pre-configured and re-packaged the base release to create the "agencyKit" package (contact me or visit the site to download it at <http://www.picNiche.com/agencykit/>). The agencyKit (built from this design) provides a set of functionality geared specifically for companies licensing digital content, primarily for microstock image websites.

By using the apiKit (agencyKit) package, you have a headstart on creating an API (not 'only' for contributors), reducing the required workload significantly.

Notes

This document does not exhaustively cover every aspect of a contributor-api project, but should suffice as an outline or guide to the implementation thereof.

This design (and the apiKit-based agencyKit framework) was built speculatively; meaning I (Bob Davies) do not own or run a business selling rights to use or access content on behalf of content owners. I have a technical history in workflow, efficiency and business-to-business software. I built this design to encourage development of contributor-focussed APIs within the stock photography industry.

This document (dependent upon the format you have received it in) may be accompanied by an API framework preconfigured with the (unimplemented) functions described above (the agencyKit), downloadable at <http://www.picNiche.com/agencykit/>. It is built upon the open-source apiKit project found at <http://www.apiKit.org> (Also built by Bob Davies, shared under a creative-commons attribution share-alike license).

Revisions

This project is open and collaborative, all contributors are welcome and I would very much appreciate your sharing this document and your thoughts on it with your colleagues, co-workers or other interested third-parties.

Please update with any revisions (if accessing the online version) or send revisions, questions, suggestions or additions to Bob Davies at admin@picNiche.com

Thank you

Thank you for taking the time to read, I hope you find the prospect of implementing systems to foster greater inter-company communication and improved crowd-sourcing processes as exciting as I do :)

Bob