Kadanza Asset Manager

API Documentation

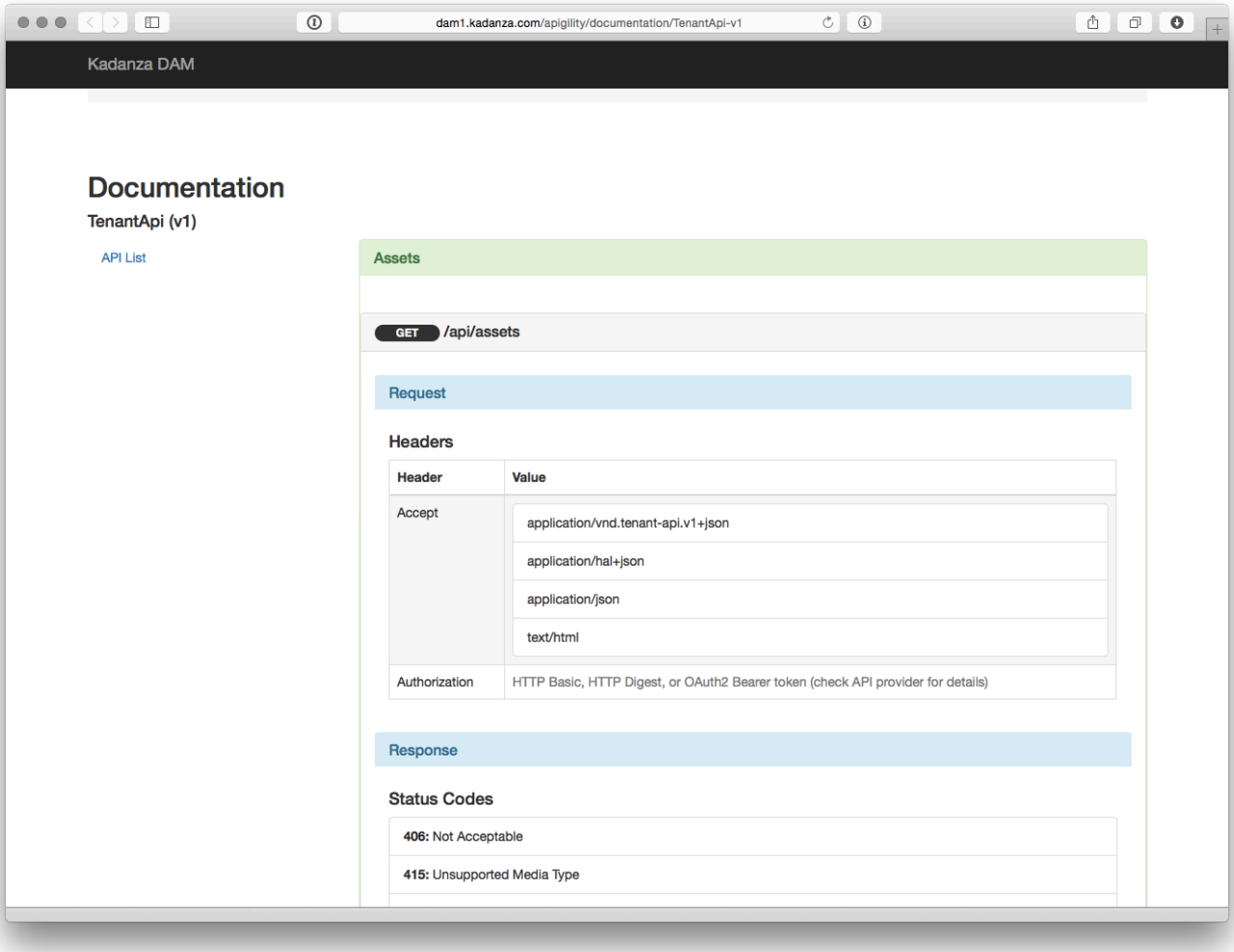
Version 1.1 - Feb 2019

KADANZA

API

In orde to use the API, every call needs to be authenticated. More information about the authentication, see: Token authentication.

The API documentation itself can be found here: http://dam1.kadanza.com/apigility/documentation/TenantApi-v1



Token authentication

Depending on the type of connection, there are different ways that an API can authenticate on the server. All types of authentication require a secret key, which can be configured per tenant.

HTTP Header

To authenticate on the server, a custom Authorization header needs to be available in the HTTP Call. The value of this header needs to be compatible with http-token-auth-01.

```
Authorization: Token token="{token-data}", auth="{hash}"
```

You need to set some parameters in this authentication string:

Token-data

This is a string that contains base_64 encoded JSON data. The available data parameterst are:

- *tenant:* The tenant username
- *type:* The type of token. Can be: `user` for client to server authentication.
- *user:* The username of the current user. This can be left `null` for tenant types.
- *expires*: The unix timestamp of the moment the token will be expired.

```
$data = array();
$data['tenant'] = 'tenantname';
$data['type'] = 'tenant';
$data['user'] = 'username';
$data['expires'] = strtotime('+1 day');
$data = array_filter($data);
ksort($data);

$tokenData = base64_encode(json_encode($data));
```

Note: Make sure to filter out empty data in the token.

Hash

This md5 hash is calculated based on the token-data and the secret key.

```
$secretKey = 'secret-tenant-key';
$hash = hash_hmac('md5', $tokenData, $secretKey);
```

Token types

Client to Server Tokens

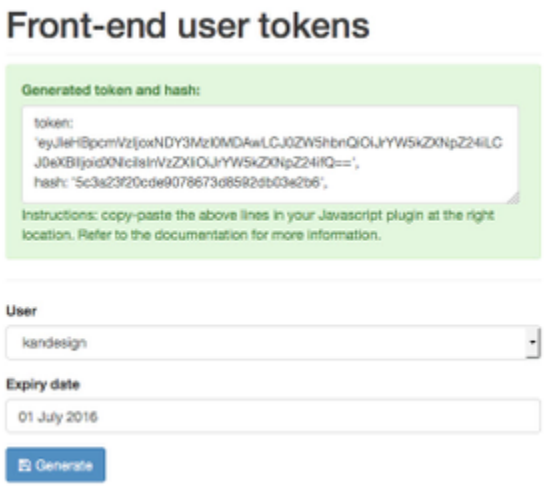
When you want to connect to the DAM API, you will need a client to server token. The current authenticated user that wants to use the DAM is included in the token-data. At this point, our server knows which user is connecting and what the permissions for the user are

Token-data:

- tenant: the tenant name
- type: user
- user: the username of the authenticated user
- expires: timestamp

Manually generate user tokens

It's possible to generate client/user tokens in de backend (use "Front-end user tokens" in the backend navigation). Select a *front-end user* and pick an *expiry date.*
After submitting the form a `token` and `auth` string/code is generated. Both can be used to configure the plugin: Javascript plugin.



Chili Hash

The integration with CHILI works a little bit different. It is not possible to add an Authorization header to the request from CHILI. Therefore a hash is added to the query of the URL. The hash is calculated as followed:

```
$userName = 'username';
$secretKey = 'secret-tenant-key';
$token = hash_hmac('md5', $userName, $secretKey);
```

In this case, you provide the username of the current authenticated user and hash it with the tenant secret key.

Javascript plugin

The application can be initialized with a jQuery plugin called `dam`. In this document you can find out how it can be used and configured.

Table of contents:

Usage

It is very easy to attach the application to a specific DOM element:

```
$('#dam-div').dam({options});
```

Note: Only one instance can be created on a single page.

Example

```
<div id="dam-div" style="position:relative;"></div>
 <script
src="http://accdam.kadanza.com/plugin/scripts/jquery.dam.js"></script>
 <script type="text/javascript">
  (function($) {
   $('#dam-div').dam(
    {
     backend: 'http://accdam.kadanza.com',
     basePath: 'http://accdam.kadanza.com/plugin',
       apiPath: '/api',
        token:
'eyJleHBpcmVzIjoxNDE1ODg3NjUxLCJ0ZW5hbnQiOiJjb2xydXl0IiwidHlwZSI6InVzZXI
iLCJ1c2VyIjoidG9vbi52ZXJ3ZXJmdCJ9',
        hash: '8852aecc1724ce2a85735ced49d2609e',
        language: 'en_EN',
     tenant: 'colruyt',
     categoryGroup: null,
     startCategory: null,
     selectedOverviewTemplate: 'icon',
     sortingSearch: {
         field: 'relevance',
         direction: 'ASC'
     },
                  sortingList: {
                      field: 'uploaded_on',
                      direction: 'DESC'
                  },
```

```
                mode: 'manager',
viewMode: 'browse',
permission: null,
format: null,
workspace: {
 thumbnails: {
  crop: true
 }
},
flags: {
 addTags: true,
 addTagsBasedOnMetadata: true
  // Add flags here
}
toolbars: {
   actionButtons: {
     display: true,
     addAsset: true,
     editAsset: true,
     removeAsset: true,
     collections: true,
     addCollection: true,
     selectAsset: true
   },
 stepMode: {
  enableBreadcrumb: true,
  startCategory: 0,
  updateMode: 'query',
  labels: {
   firstStep: 'Label 1',
   secondStep: 'Label 2',
   thirdStep: 'Label 3',
   button: 'Label 2' }
 },
   search: {
     display: true
   },
   sort: {
     display: true
   },
   categories: {
  displayEmptyCategories: true,
  display: true,
     open: false,
     browse: false,
  closeOnSelect: false,
    breadcrumbHomeText: 'Home'
 },
   assets: {
     report: false
```

```
      },
      overviewTypes: {
        display: true
      },
      tags: {
        multiSelect: true
      }
    }
  }
);
```

```
  })(jQuery);
 </script>
</div>
```

Parameters

Parameter	Type	Default	Description	Notes
backend	String	http://dam.kadanza.local	The URL to the back-end server.	
basePath	String	(empty)	The base path for loading the CSS and JS files during initialization. This option will mostly just be empty.	
apiPath	String	/api	The path to the API on the back-end server.	
token	String	(empty)	The API authentication token. More information about the token/hash can be found at: Token authentication - Client To Server tokens.	
hash	String	(empty)	The API authentication hash. More information about the token/hash can be found at: Token authentication - Client To Server tokens.	
skipDependencyCheck	Boolean	false	This option can be set to skip the dependency injection. When disabled, you should provide all the required packages yourself. This option should always be false on PROD and ACC. It can be true in development mode to overwrite some depencencies localy.	
language	en_EN	en_EN	A valid ISO language code (ISO-369). When the selected language is not available, there will be a fallback to `en_EN`. Available language codes are: `en_EN`, `nl_BE`, `fr_BE`, `es_ES`.	
tenant	String	(empty)	This field is required. The plugin will not be able to initialize without a tenant.	
categoryGroup	String	null	The category group id you want to load. When no category group is provided, all assets will be displayed.	
startCategory	Integer	null	The catagoy id you want to open the plugin with: the assets of this category will be loaded and the start categoy is active in the category-breadcrumb. This property will be overruled when category URL query parameter is available, for example: domain.com/test-plugin#/overview?categoy=3.	
selectedOverviewTemplate	String	icon	The overview template you want to use as default. May be `'list'` or `'icon'`.	
sortingSearch > field	String	relevance	The sorting field after a search. The relavance property is handled by Elastic Search itself. May be `'relevance'`, `'title.raw'`, `'filename.raw'`, `'format'`, `'uploaded_on_raw'`, `'updated_on_raw'`, `'colorspace'`, `'resolution'`, `'orientation'`, `'width'` or `'height'`.	
sortingSearch > direction	String	ASC	The sorting direction after a search. May be `'ASC'` or `'DESC'`.	
sortingList > field	String	uploaded_on_raw	The sorting field during a list action without search query. May be `'title.raw'`, `'filename.raw'`, `'format'`, `'uploaded_on_raw'`, `'updated_on_raw'`, `'colorspace'`, `'resolution'`, `'orientation'`, `'width'` or `'height'`.	
sortingList > direction	String	DESC	The sorting direction during a list action without search query. May be `'ASC'` or `'DESC'`.	
mode	String	manager	The mode in which the asset manager is operating. May be 'manager' or 'picker'. This property will change the asset action buttons.	
viewMode	String	browse	Viewmode in which the asset manager is operating. May be 'browse' or 'step'. This property will activate Step-mode	
format	String	(empty)	This property can contain a file format on which the asset list will be filtered. File formats are listed like this: 'psd, jpg'	

permission	Sting	(empty)	This property can contain a permission on which the asset list will be filtered. It only show the assets where the current has that permission on those assets. This is used for example in combination with the CHILI Asset Picker, there you only want to show assets where the current user has the "asset.download-original" permission. Note: Of course the user also needs asset.view permission on the assets.	
workspace > thumbnails > crop	Boolean	true	Enables the cropping of thumbnails	
workspace > toolbars > actionButtons > display	Boolean	true	Display buttons toolbar. When this option is disabled, the actionButtons will NEVER be displayed. Even if they are enabled with button specific options.	
workspace > toolbars > actionButtons > addAsset	Boolean	true	Display add asset buttons.	
workspace > toolbars > actionButtons > editAsset	Boolean	true	Display edit asset buttons.	
workspace > toolbars > actionButtons > removeAsset	Boolean	true	Display remove buttons.	
workspace > toolbars > actionButtons > collections	Boolean	true	Display collections buttons.	
workspace > toolbars > actionButtons > addCollection	Boolean	true	Display add to collection buttons.	
workspace > toolbars > actionButtons > selectAsset	Boolean	true	Display select asset buttons.	
workspace > toolbars > search > display	Boolean	true	Display search toolbar	
workspace > toolbars > sort > display	Boolean	true	Display sort toolbar	
workspace > toolbars > categories > displayEmptyCategories	Boolean	true	Show all categories, including categories which are empty. When set to `false`, all empty categories will be hidden and excluded from the dropdown list. Be carefull with this option, because the backend will check every subcategory until deepest sub-level. This check stops when at least one asset is found.	
workspace > toolbars > categories > display	Boolean	true	Display categories toolbar	
workspace > toolbars > categories > open	Boolean	true	Open the categories toolbar by default	
workspace > toolbars > categories > browse	Boolean	false	Enables the category browser in the assets list	
workspace > toolbars > categories > closeOnSelect	Boolean	false	When this option is enabled, the category dropdown will be closed after the user selects (click) a category.	
workspace > toolbars > categories > breadcrumbHomeText	String	Home	The text of the first item of the category breadcrumb. Default "Home".	
workspace > toolbars > overviewTypes > display	Boolean	true	Display overview types toolbar	
workspace > toolbars > assets > report	Boolean	false	Makes it possible to report an issue for users with view rights	
workspace > toolbars > tags > multiSelect	Boolean	true	Makes it possible to enable or disable the multi-select component for tags	
workspace > toolbars > stepMode > enableBreadcrumb	Boolean	true	Enable the breadcrumb while in Step-mode.	
workspace > toolbars > stepMode > startCategory	Integer	0	Pick which category acts as root for Step-mode.	

workspace > toolbars > stepMode > updateMode	String	query	Determines when results get updated in step-mode. Valid options are 'query' and 'filter'. 'query' will update results after the user went through all steps. 'filter' will update results as soon as any dropdown changes.	
workspace > toolbars > stepMode > labels > firstStep	String	Label 1	Label text for the first step.	
workspace > toolbars > stepMode > labels > secondStep	String	Label 2	Label text for the second step.	
workspace > toolbars > stepMode > labels > thirdStep	String	Label 3	Label text for the third step.	
workspace > toolbars > stepMode > labels > button	String	Label button	Label text for the update results button.	
flags > addTags	Boolean	true	When this option is disabled, the user will not be able to attach new tags to assets. The user will be restricted to use the tags that are available in the back-end. Options: "true", "false"	
flags > addTagsBasedOnMetadata	Boolean	true	When this option is enabled, the backend creates new tags based on IPTC/XMP metadata of the uploaded asset. If disabled, the tags from IPTC/XMP metadata will be ignored. Options: "true", "false"	
flags > debug	Boolean	false	This option will run the application in debug mode. Some extra features like improved error logging will be activated.	
taskManager > poller > interval	Integer	5000	The amount of milliseconds the TaskManager waits to check if a task has completed.	
taskManager > poller > maxAttempts	Integer	100	The maximum amount of times the TaskManager will check if a task has completed.	
assets > report	Boolean	false	Enables "Asset reportorting" feature in the preview modal of an asset.	

Events

Depending on your configuration, there is a list of events that are being triggered by the application. Listening to events:

```
$('#dam-div').on('[event]', function(event, data) {
    // your callback method
    var plugin = data.plugin;
});
```

cssLoaded

This event is triggered when all CSS files are loaded. Most likely you want to attach your custom CSS files when this event is triggered. Also take a look at the chapter about the File loader to make it easy to load your files.

checkedBrowserCompatibility

This event is triggered when the browser compatibility has been checked. At this point the application should work browser independent.

dependenciesLoaded

This event is triggered when all javascript dependencies are loaded. You can load custom Javascript files at this point. Also take a look at the chapter about the File loader to make it easy to load your files.

appInitialized

This event is triggered when the application is fully initialized and working as suspected.

File loader

In the plugin there is a file loader available. This can be handy to require JS and CSS files on a specific event:

```
var fileLoader = plugin.fileLoader;
var useBasePath = false;
var location;
location = fileLoader.getFileLocation('/script/script.js', useBasePath);
fileLoader.loadJs(location);
location = fileLoader.getFileLocation('/styles/styles.js', useBasePath);
fileLoader.LoadCss(location);
```

Post-messages

When the Asset Manager is included in an iframe, it is possible to interact with it through post messages.

Currently following post messages are handled / sent:

sender	receiver	type	Sample response	Trigger
asset-manager	parent	asset-picked	{"type":"asset-picked","url":"http://link.to/image.jpg ", "asset":{"id":123,"fileName":"unique_filename.jpg"}}	This message is triggered by the asset-manager when a user clicks on the 'Pick this asset' button in picker modus.

Sending messages from a parent window to the asset-manager iframe:

```
var message = JSON.stringify({type: "typename", property: "value"});
$('iframe').get(0).contentWindow.postMessage(message, "*");
```

Listening to messages from the asset manager in the iframe:

```
var callback =  function (e) {
 var message = JSON.parse(e.data);
};

if (window.addEventListener) {
    // Normal browsers
    window.addEventListener('message', callback, false);
} else {
    // IE 8
    window.attachEvent('onmessage', callback);
}
```