Daminion API

Authentication

Request

Method:

POST

Endpoint:

/account/login

Headers:

Content-Type: application/json

Body: {

"usernameOrEmailAddress": **string**,

Username

"password": **string**

User password

}

Example:

```
POST /account/login HTTP/1.1
Host: test.daminion.net
Content-Type: application/json
Content-Length: 69

{
    "usernameOrEmailAddress": "admin",
    "password": "admin"
}
```

Response

Headers:

`Set-Cookie: .AspNet.ApplicationCookie={cookie}`

This response header returns an identification cookie that must be added to the headers of any requests that require authorization. If authentication fails, this header is missing.

Getting Tags

Request

Method:

`GET`

Endpoint:

`/api/settings/getTags`

Headers:

`Cookie: .AspNet.ApplicationCookie={cookie}`

Identification cookie received after the authentication

Example:

```
GET /api/settings/getTags HTTP/1.1
Host: test.daminion.net
Cookie:
.AspNet.ApplicationCookie=aoHzhsQUMgPtZbm3tDixCHSEqK4ZxqvAY4Qg
ZX3jrqxvqXkjViLxpd39bk_FV4NRd6ViLoCQrQ3noa6YIjfTwKTIdCvFkTOPXf
YXXH8LY2hIMlJGaBIkoKs2Mk16H-_bYYDkmmtgfOj4F57KzaN_mQG0NXf-fJub
Edpgi9aQ29Vo5RC9KgRfOEBcrJbPq-nrwdmhVzVqX_5SiEkvsIzcoKA-AVAYCC
bWtYcGPOVqDztQDdXGJk-4Am1mxxEq6NqEVrmq8vv3EmvE6nAO04zLX3FBs0no
rabaxYJMOTLScXDBX2LhWqTWXOi_-rLhomGzY3o4zo7rcDJ227jFP7mHIotjUy
bNPQaoQKXb9__K24YQdRdgMWQi77KPF-QmUeNiPr3he0YZTRSEfG9c7cFnbeLT
lW-YjSOwPEuWNKZRTFE
```

Response

Headers:

```
Content-Type: application/json; charset=utf-8
```

Body: {

“data”: **array(**Tag**)**,

Array of the tags. The tag’s structure is described below.

“error”: **string**,

Error description.

“errorCode”: **number**,

Error code.

“success”: **boolean**

The status of the successful completion of the search.

}

Tag: {

"id": **number**,

Tag ID.

"indexed": **boolean**,

Tag type. If the value is true, the tag is indexed, otherwise the tag is linear.

"guid": **string**,

Tag GUID.

"name": **string**,

Current tag name.

"originName": **string**,

Original tag name (e.g. prior to renaming).

"readOnly": **boolean**,

If the value is true, new tag values cannot be created and existing tag values cannot be edited or deleted.

"dataType": **string**

Tag data type.

}

Creating a custom tag

Request

Method:

`POST`

Endpoint:

`/api/indexedTagValues/createCustomTag`

Headers:

`Content-Type: application/json`

`Cookie: .AspNet.ApplicationCookie={cookie}`

Identification cookie received after the authentication.

Body: {

`"name":` **`string`**,

Tag name. ***The name should be unique!*** *The system should not contain two tags with the same name to avoid collisions!*

`"type":` **`number`**,

Data type. Number from 0 to 4:

0 – string

1 – fractional

2 – integer

3 – date and time

4 – currency

`"multiplyValues":` **`boolean`**,

Flag that defines if multiple tag values can be assigned to the file. For example, multiple values of the tag Categories can be assigned to the file. ***NOTE: If value is set to True, an indexed tag will be created****.*

`"hierarchy":` **`boolean`**,

Flag that defines if hierarchy of tag values can be created. Tag values can have child values. ***NOTE: If value is set to True, an indexed tag will be created****.*

`"allowSynonyms":` **`boolean`**,

Flag that allows the use of synonyms for tag values.

"limitedNumber": **boolean**

Flag that defines that only one tag value can be assigned to the file. For example, only one value of the tag Color Label can be assigned. ***NOTE: If value is set to True, an indexed tag will be created****.*

}

Example:

```
POST /api/indexedTagValues/createCustomTag HTTP/1.1
Host: test.daminion.net
Content-Type: application/json
Cookie:
.AspNet.ApplicationCookie=aoHzhsQUMgPtZbm3tDixCHSEqK4ZxqvAY4Qg
ZX3jrqxvqXkjViLxpd39bk_FV4NRd6ViLoCQrQ3noa6YIjfTwKTIdCvFkTOPXf
YXXH8LY2hIMlJGaBIkoKs2Mk16H-_bYYDkmmtgfOj4F57KzaN_mQG0NXf-fJub
Edpgi9aQ29Vo5RC9KgRfOEBcrJbPq-nrwdmhVzVqX_5SiEkvsIzcoKA-AVAYCC
bWtYcGPOVqDztQDdXGJk-4Am1mxxEq6NqEVrmq8vv3EmvE6nAO04zLX3FBs0no
rabaxYJMOTLScXDBX2LhWqTWXOi_-rLhomGzY3o4zo7rcDJ227jFP7mHIotjUy
bNPQaoQKXb9__K24YQdRdgMWQi77KPF-QmUeNiPr3he0YZTRSEfG9c7cFnbeLT
lW-YjSOwPEuWNKZRTFE
Content-Length: 154

{
    "name": "Company",
    "type": 0,
    "multiplyValues": false,
    "hierarchy": true,
    "allowSynonyms": false,
    "limitedNumber": true
}
```

Getting tag values

Request

Method:

`GET`

Endpoint:

`/api/indexedTagValues/getIndexedTagValues`

Query parameters:

`indexedTagId`

ID of the tag which values should be found.

`pageSize`

Page size. A positive integer no greater than 2 147 483 647.

`pageIndex`

Page serial number. An integer from 0 to 2 147 483 647 (inclusive).

`parentValueId`

Parameter to limit the search level. This parameter can take special values or tag value ID. The following special values are possible:

0 – searching for values only at the root level (the search of child values will not be performed in this case)
-2 – throughout search (the search will be performed at the root level and among the child values)

If the parameter sends the ID of any tag value that has child values, the search will be performed only among the child values of the first level.

`filter`

Case-insensitive search string to filter values. This parameter can take an empty value. If the string is filled in, only those values will be found whose text representation contains the searched string.

Headers:

`Cookie: .AspNet.ApplicationCookie={cookie}`

Identification cookie received after the authentication

Example:

```
GET
/api/indexedTagValues/getIndexedTagValues?filter=&pageSize=16&
indexedTagId=76&pageIndex=0&parentValueId=0 HTTP/1.1
Host: test.daminion.net
Cookie:
.AspNet.ApplicationCookie=aoHzhsQUMgPtZbm3tDixCHSEqK4ZxqvAY4Qg
ZX3jrqxvqXkjViLxpd39bk_FV4NRd6ViLoCQrQ3noa6YIjfTwKTIdCvFkTOPXf
YXXH8LY2hIMlJGaBIkoKs2Mk16H-_bYYDkmmtgfOj4F57KzaN_mQG0NXf-fJub
Edpgi9aQ29Vo5RC9KgRfOEBcrJbPq-nrwdmhVzVqX_5SiEkvsIzcoKA-AVAYCC
bWtYcGPOVqDztQDdXGJk-4Am1mxxEq6NqEVrmq8vv3EmvE6nAO04zLX3FBs0no
rabaxYJMOTLScXDBX2LhWqTWXOi_-rLhomGzY3o4zo7rcDJ227jFP7mHIotjUy
bNPQaoQKXb9__K24YQdRdgMWQi77KPF-QmUeNiPr3he0YZTRSEfG9c7cFnbeLT
lW-YjSOwPEuWNKZRTFE
```

Response

Headers:

```
Content-Type: application/json; charset=utf-8
```

Body: {

“values”: **array(**TagValue**)**,

Array of tag values. The structure of tag values is described below.

“path”: **array(**TagValue**)**,

Array of parent tag values, where the first element is the parent located at the root level and the last element is the immediate current parent.

“tag”: **Tag**,

An object containing information about the tag where the search is performed. The structure of the tag is described below.

“error”: **string**,

Error description.

“errorCode”: **number**,

Error code.

"success": **boolean**

The status of the successful completion of the search

}

TagValue: {

"text": **string**,

Text representation of the tag value.

"id": **number**,

Tag value ID.

"isDefaultValue": **boolean**,

If true, then this is a system value, otherwise the value is custom. An example of a system tag value is the value Unassigned. This value is created automatically and cannot be changed or deleted.

"tagId": **number**,

ID of the tag where a tag value is created.

"rawValue": **string**,

System representation of the tag value. Depending on a tag, it may be the same as the text representation, but it may also be a different, special value.

"tagName": **string**,

The name of the tag for which the value was created.

"hasChilds": boolean

If true, then the value has child values.

}

Tag: {

"id": **number**,

Tag ID.

"indexed": **boolean**,

Tag type. If true, then this tag is indexed. Otherwise, the tag is linear.

"guid": **string**,

Tag GUID.

"name": **string**,

Current tag name.

"originName": **string**,

Original tag name (e.g. prior to renaming).

"readOnly": **boolean**,

If the value is true, new tag values cannot be created and existing tag values cannot be edited or deleted.

"dataType": **string,**

Tag data type.

"isAllowAssign": **boolean**,

If true, then the tag values of this tag can be assigned to media items.

"maxHierarchy": **number**,

The maximum allowed nesting depth for the tag.

"strongHierarchy": **boolean**,

Flag for using a strict hierarchy of tag values. There are a few tags with a predefined hierarchy structure. For example, the Place tag can have a maximum of 4 levels of hierarchy and each level will always have at least one value.

"isMultiplyValues": **boolean**,

If true, then multiple tag values can be assigned to the same media item. This relationship is also called many-to-many.

"allowSearch": **boolean**

If true, then the tag value can be filtered using the parameter filter.

}

Creating a tag value

Request

Method:

POST

Endpoint:

/api/indexedTagValues/createValueByGuid

Headers:

Content-Type: application/json

Cookie: .AspNet.ApplicationCookie={cookie}

Identification cookie received after the authentication.

Body: {

"guid": **string**,

The GUID of the tag for which the new tag value is created.

"parent": **number,**

The ID of the parent value. Optional field that can be omitted if the root value is created, or the tag for which the value is created is not hierarchical.

"value": **string,**

Text representation of the tag value. For hierarchical tags, it is possible to specify a value separated by backslashes to indicate parent values. For example, if you pass a string like "Animal\\\Cat", two values will be created at once: the root value "Animal" and the child value "Cat".

}

Example:

```
POST /api/indexedTagValues/createValueByGuid HTTP/1.1
Host: test.daminion.net
Content-Type: application/json
Cookie:
.AspNet.ApplicationCookie=aoHzhsQUMgPtZbm3tDixCHSEqK4ZxqvAY4Qg
ZX3jrqxvqXkjViLxpd39bk_FV4NRd6ViLoCQrQ3noa6YIjfTwKTIdCvFkTOPXf
YXXH8LY2hIMlJGaBIkoKs2Mk16H-_bYYDkmmtgfOj4F57KzaN_mQG0NXf-fJub
Edpgi9aQ29Vo5RC9KgRfOEBcrJbPq-nrwdmhVzVqX_5SiEkvsIzcoKA-AVAYCC
```

```
bWtYcGPOVqDztQDdXGJk-4Am1mxxEq6NqEVrmq8vv3EmvE6nAO04zLX3FBs0no
rabaxYJMOTLScXDBX2LhWqTWXOi_-rLhomGzY3o4zo7rcDJ227jFP7mHIotjUy
bNPQaoQKXb9__K24YQdRdgMWQi77KPF-QmUeNiPr3he0YZTRSEfG9c7cFnbeLT
lW-YjSOwPEuWNKZRTFE
Content-Length: 297

{
    "guid": "b855732d-7747-464b-9712-cc977c2f4da6",
    "parent": 73,
    "value": ”Car\\BMW”
}
```

Response

Headers:

```
Content-Type: application/json; charset=utf-8
```

Body: {

“data”: **array(**TagValue**)**,

An array containing information about the created tag values (including parent values).

“error”: **string**,

Error description.

“errorCode”: **number**,

Error code.

“success”: **boolean**

The status of the successful completion of the search

}

TagValue: {

"text": **string**,

Text representation of the tag value.

"id": **number**,

Tag value ID.

"isDefaultValue": **boolean**,

If true, then this is a system value, otherwise the value is custom. An example of a system tag value is the value Unassigned. This value is created automatically and cannot be changed or deleted.

"tagId": **number**,

ID of the tag where a tag value is created.

"rawValue": **string**,

System representation of the tag value. Depending on a tag, it may be the same as the text representation, but it may also be a different, special value.

"tagName": **string**,

The name of the tag for which the value was created.

"hasChilds": boolean

If true, then the value has child values.

}

Getting sort tags

Request

Method:

GET

Endpoint:

/api/mediaItems/getSort

Headers:

Cookie: .AspNet.ApplicationCookie={cookie}

Identification cookie received after the authentication

Example:

```
GET /api/mediaItems/getSort HTTP/1.1
Host: test.daminion.net
Cookie:
.AspNet.ApplicationCookie=aoHzhsQUMgPtZbm3tDixCHSEqK4ZxqvAY4Qg
```

```
ZX3jrqxvqXkjViLxpd39bk_FV4NRd6ViLoCQrQ3noa6YIjfTwKTIdCvFkTOPXf
YXXH8LY2hIMlJGaBIkoKs2Mk16H-_bYYDkmmtgfOj4F57KzaN_mQG0NXf-fJub
Edpgi9aQ29Vo5RC9KgRfOEBcrJbPq-nrwdmhVzVqX_5SiEkvsIzcoKA-AVAYCC
bWtYcGPOVqDztQDdXGJk-4Am1mxxEq6NqEVrmq8vv3EmvE6nAO04zLX3FBs0no
rabaxYJMOTLScXDBX2LhWqTWXOi_-rLhomGzY3o4zo7rcDJ227jFP7mHIotjUy
bNPQaoQKXb9__K24YQdRdgMWQi77KPF-QmUeNiPr3he0YZTRSEfG9c7cFnbeLT
lW-YjSOwPEuWNKZRTFE
```

Response

Headers:

```
Content-Type: application/json; charset=utf-8
```

Body: {

“items”: **array(**SortTag**)**,

Array of tags to sort. The structure is described below.

“error”: **string**,

Error description.

“errorCode”: **number**,

Error code.

“success”: **boolean,**

The status of the successful completion of the search.

"totalCount": **number**

The total number of tags to sort.

}

SortTag: {

"name": **string**,

Tag name.

"id": **number**

Tag ID.

}

Search for media items

Request

Method:

`GET`

Endpoint:

`/api/mediaItems/get`

Query parameters:

`queryLine`

Search conditions separated by a semicolon and set in a special format (a detailed description of the format is given in the section “Parameter ‘query line’”). This is an optional parameter. If it is not set, all media items available for this user will be found.

`f`

Logical operators separated by a semicolon and set in a special format (a detailed description of the format is given in the section “Logical operators”) . This is an optional parameter that can be omitted.

`size`

Page size. Can take positive integer values from 0 to 1000.The parameter is optional. If the parameter is not specified, the page size is 0.

`index`

The serial number of the requested page. Integer between 0 and 2147483647 (inclusive). The parameter is optional. If the parameter is not specified, index is 0.

`sortTag`

Tag ID to sort by. A limited number of tags can be used for sort (additional information is given in the section “Getting tags for sort”). This is an optional parameter. If it is not set, the sort is not applied.

`sort`

Sort order. Can take one of two values: ASC (ascending) and DESC (descending). This is an optional parameter. If it is not set, ascending order is used in sort.

Headers:

`Cookie: .AspNet.ApplicationCookie={cookie}`

Identification cookie received after the authentication

Example:

```
GET
/api/mediaItems/get?queryLine=5%2C9%3B76%2C12%3B76%2C5&f=5%2Ca
ny%3B76%2Cany&index=0&size=70&sort=ASC&sortTag=5002 HTTP/1.1
Host: test.daminion.net
Cookie:
.AspNet.ApplicationCookie=aoHzhsQUMgPtZbm3tDixCHSEqK4ZxqvAY4Qg
ZX3jrqxvqXkjViLxpd39bk_FV4NRd6ViLoCQrQ3noa6YIjfTwKTIdCvFkTOPXf
YXXH8LY2hIMlJGaBIkoKs2Mk16H-_bYYDkmmtgfOj4F57KzaN_mQG0NXf-fJub
Edpgi9aQ29Vo5RC9KgRfOEBcrJbPq-nrwdmhVzVqX_5SiEkvsIzcoKA-AVAYCC
bWtYcGPOVqDztQDdXGJk-4Am1mxxEq6NqEVrmq8vv3EmvE6nAO04zLX3FBs0no
rabaxYJMOTLScXDBX2LhWqTWXOi_-rLhomGzY3o4zo7rcDJ227jFP7mHIotjUy
bNPQaoQKXb9__K24YQdRdgMWQi77KPF-QmUeNiPr3he0YZTRSEfG9c7cFnbeLT
lW-YjSOwPEuWNKZRTFE
```

Response

Headers:

`Content-Type: application/json; charset=utf-8`

Body: {

`“error”: **string**,`

Error description.

`“errorCode”: **number**,`

Error code.

`“mediaItems”: **array**(Item),`

Array of media items. The structure of the media item is described below.

`“success”: **boolean**`

The status of the successful completion of the search.

}

Item: {

"id": **number**,

Media item ID.

"hashCode": **number**,

Media item hash which is recalculated when a file or a new file version is imported.

"name": **string**,

Media item name. This field will contain one of the tags: Description, Title or Filename.

"fileName": **string**,

File name of the media item.

"mediaFormat": **string**,

File format of the media item.

"versionControlState": **number**,

File status. The following values are possible:

0 – the file is available (not checked out by any of the users).

1 – locked (checked out) by this user.

2 – locked (checked out) by another user.

"colorLabel": **number**,

ID of the value of the tag "Color Label".

"width": **number**,

Horizontal resolution of the media item. If the resolution of the current media item cannot be determined, the value will be set to 0.

"height": **number**,

Vertical resolution of the media item. If the resolution of the current media item cannot be determined, the value will be set to 0.

"fileSize": **number**,

File size in bytes.

"formatType": **string**,

Media item format.

"expirationDate": **string**

License expiration date of the file in ISO 86011 format.

}

Parameter ‘query line’

Depending on the tag type, the search can be performed by the tag value ID, part of the string representation of the tag value, or a range of values.

Search by tag value ID

This type of search is only possible with indexed tags. The search condition is passed as a string containing the tag ID and the tag value ID, separated by a comma. Getting tag IDs and tag value IDs is described in the sections “Gettings tags” and “Getting tag values”.

For example, if the tag “Color Label” has the ID “5”, and its tag value “Brown” has the ID “8”, then to search for all media items that have the tag value “Brown”, the following condition should be used:

```
queryLine=5,8
```

The search conditions by ID may contain the operand NOT to search for all media items that are not assigned to a specific tag value. To use negation, an exclamation mark (symbol “!”) should be added before the ID of this tag value.

For example, to find all the media items that are not assigned to “Brown”, the following search condition should be used:

```
queryLine=5,!8
```

Search by a string representation of the tag value

This type of search is possible for:

- any linear tag of the type “Text” and “Link”;
- any custom indexed tag of the type “Text”
- certain system indexed tags: Folders, Color Label, People, Place, Event, Categories, Auto Tags, Keywords, Project, Client, Authors, Copyrights, Scene, Subject Code, Intellectual Genre, Color Profile, Fonts, Camera Model, Video Codec, Artists, Flag, Shared Collections
- tag “Everywhere” - this is a virtual tag with the ID 5000 that includes the following tags: Filename, Folder, Color Label, Title, Description, Categories, Collections, Keywords, Auto Tags, Place, Event, Authors, Copyrights, People, Color Profile, Camera Vendor, Camera Model, Camera Lens, Project, Client, Content.

The result of this search will be a list of media items that have at least one tag value containing the search string in any part of the text representation of the tag value.

The search condition is passed as a string that contains the tag ID and a case-insensitive text representation of the tag value, separated by a comma.

For example, the linear tag “Filename” has the ID “5001”. To find all media items with the filename containing the word “design”, the following condition should be used:

```
queryLine=5001,design
```

Search by a range of values

This type of search is only available for linear tags of types “Fractional”, “Integer”, “Time” and “Currency”. The search condition is passed as a string containing the tag ID and a special representation of the required search range:

```
queryLine={tagId},([{start}])-([{end}])
```

tagId – Id of the tag to be used in the search condition

start – start value of the search range (inclusive).

end – end value of the search range (inclusive).

At least one value of the search range should be given. If the start value of the search range is not given, the media items with a tag value less than or equal to the end value of the range will be selected. If the end value of the search range is not given, the media items with a tag value greater than or equal to the start value of the range will be selected.

To specify the values of the search range, integer and fractional numbers (a period must be used for the decimal separator), as well as a date (in ISO 8601 format) can be used.

For example, if the linear tag “Width” has the ID “5018” and it is required to find all media items with the width of 1900 px and greater, the following condition should be used:

```
queryLine=5018,(1900)-()
```

Combined conditions

Oftentimes, it is required to perform a search by multiple conditions.
For example, it is required to find all images with the “Red” Color Label and the license for use that is not expired. In fact, this is a combination of conditions by tags Media Format, Color Label and License Expiration Date. In the queryLine parameter, multiple conditions can be used and separated by semicolon (symbol “;”).

For example, the tags Media Format, Color Label and License Expiration Date have the IDs “2”, “5” and “5006” respectively. The tag value “Images” of the tag Media Format has the ID “1”. The current date is 2022-04-06T16:56:55.044Z. To perform the search, the parameter queryLine can be passed as follows:

```
queryLine=2,1;5,Red;5006,(2022-04-06T16:56:55.045Z)-()
```

Logical operators

The use of logical operators comes into play if you want to search multiple tag value IDs of a multiple indexed tag. A multiple indexed tag is a tag whose different values can be assigned to the same media item. A good example of a multiple indexed tag is the tag Categories.

Two types of logical operators are supported: all (logical AND) and any (logical OR). The logical operators are passed as a string containing a tag ID and a logical operator separated by a comma:

```
f={tagId},all|any
```

For example, the tag Categories with the ID “10” has the tag values “Car” and “Driver” with the IDs “6” and “15” respectively. It is required to find all media items with the tag values “Car” and “Driver”. To do this, the following queryLine parameters should be passed:

```
queryLine=10,6;10,15&f=10,all
```

As the result of the query with these parameters, all media items with the tag values “Car” and “Driver” will be found.

If all media items with one of two tag values should be found, the operator “any” should be used.

```
queryLine=10,6;10,15&f=10,any
```

Different logical operators can be combined for different tags. To do this, multiple logical operators should be separated by a semicolon (symbol “;”).

For example, it is required to find all media items with the tag values “Car” and “Driver” of the tag Categories and the tag value “John” or “Steve” of the tag People. If the tags People has the ID “7” and tag values “John” and “Steve” have the tag value IDs “75” and “82” respectively, the query should contain the following parameters:

```
queryLine=7,75;7,82;10,6;10,15&f=7,any;10,all
```

Getting the absolute path to media items

Request

Method:

`GET`

Endpoint:

`/api/mediaItems/getAbsolutePaths`

Query parameters:

`ids`

A list of media items IDs separated by a comma.

Headers:

`Cookie: .AspNet.ApplicationCookie={cookie}`

Identification cookie received after the authentication

Example:

```
GET /api/mediaItems/getAbsolutePaths?ids=1,2,3,4,5 HTTP/1.1
Host: test.daminion.net
Cookie:
.AspNet.ApplicationCookie=aoHzhsQUMgPtZbm3tDixCHSEqK4ZxqvAY4Qg
ZX3jrqxvqXkjViLxpd39bk_FV4NRd6ViLoCQrQ3noa6YIjfTwKTIdCvFkTOPXf
YXXH8LY2hIMlJGaBIkoKs2Mk16H-_bYYDkmmtgfOj4F57KzaN_mQG0NXf-fJub
Edpgi9aQ29Vo5RC9KgRfOEBcrJbPq-nrwdmhVzVqX_5SiEkvsIzcoKA-AVAYCC
bWtYcGPOVqDztQDdXGJk-4Am1mxxEq6NqEVrmq8vv3EmvE6nAO04zLX3FBs0no
rabaxYJMOTLScXDBX2LhWqTWXOi_-rLhomGzY3o4zo7rcDJ227jFP7mHIotjUy
bNPQaoQKXb9__K24YQdRdgMWQi77KPF-QmUeNiPr3he0YZTRSEfG9c7cFnbeLT
lW-YjSOwPEuWNKZRTFE
```

Response

Headers:

`Content-Type: application/json; charset=utf-8`

Body: {

“**number**”: **string**,

A pair where the key is the media item's identifier and the value is the absolute path to the media item.

}

Assigning Tags

Request

Method:

POST

Endpoint:

/api/itemData/batchChange

Headers:

Content-Type: application/json

Cookie: .AspNet.ApplicationCookie={cookie}

Identification cookie received after the authentication.

Body: {

"ids": **array(number)**,

Identifiers of media objects for which the operations should be performed.

"data": **array(**Operation**)**

Array of operations that should be performed with specified media objects. The structure of the operation is described below.

}

Operation: {

"guid": **string**,

GUID of the tag which values should be assigned or deleted.

"id": **number,**

ID of the tag value which should be assigned or deleted. If the text representation of the tag value should be assigned, this field should be set to 0.

"value": **string,**

Text representation of the tag value that should be assigned. ***The value of this field is not used in the delete operation.***

"remove": **boolean**

Flag that defines the type of operation. If the field is set to true, the tag value is deleted by its ID, otherwise it is assigned by its ID or text representation.

}

Example:

```
POST /api/itemData/batchChange HTTP/1.1
Host: test.daminion.net
Content-Type: application/json
Cookie:
.AspNet.ApplicationCookie=aoHzhsQUMgPtZbm3tDixCHSEqK4ZxqvAY4Qg
ZX3jrqxvqXkjViLxpd39bk_FV4NRd6ViLoCQrQ3noa6YIjfTwKTIdCvFkTOPXf
YXXH8LY2hIMlJGaBIkoKs2Mk16H-_bYYDkmmtgfOj4F57KzaN_mQG0NXf-fJub
Edpgi9aQ29Vo5RC9KgRfOEBcrJbPq-nrwdmhVzVqX_5SiEkvsIzcoKA-AVAYCC
bWtYcGPOVqDztQDdXGJk-4Am1mxxEq6NqEVrmq8vv3EmvE6nAO04zLX3FBs0no
rabaxYJMOTLScXDBX2LhWqTWXOi_-rLhomGzY3o4zo7rcDJ227jFP7mHIotjUy
bNPQaoQKXb9__K24YQdRdgMWQi77KPF-QmUeNiPr3he0YZTRSEfG9c7cFnbeLT
lW-YjSOwPEuWNKZRTFE
Content-Length: 410

{
    "ids": [
        1,
        2,
        3
    ],
    "data": [
        {
            "guid": "b855732d-7747-464b-9712-cc977c2f4da6",
            "id": 0,
```

```
            "value": "Car",
            "remove": false
        },
        {
            "guid": "f3dc7b01-24f3-49b4-93d2-5183c677984b",
            "id": 1,
            "value": "Red",
            "remove": false
        }
    ]
}
```

File upload

Request

Method:

`POST`

Endpoint:

`/api/import/uploadFile`

Headers:

`Content-Type: multipart/form-data`

`Cookie: .AspNet.ApplicationCookie={cookie}`

Identification cookie received after the authentication.

Body:

`file`

The file to be uploaded.

`tagValues`

A string containing a serialized JSON array. This array stores objects of the ***AssignmentData*** *type (type structure is described below).*

`AssignmentData:` {

`"guid":` **`string`**,

GUID of the tag which values should be assigned.

`"values:` **`array(string),`**

An array of string values to be assigned. Note that regardless of the tag's data type, the value must always be a string. To pass a value of the ***DateTime*** *type you must perform a conversion to format: YYYY.MM.DD.HH.MM.SS, where:*

YYYY – year

MM – month

DD – day

HH – hours

MM – minutes

SS – seconds

For example, February 13th, 2022 must be converted to:

“2022.02.13.00.00.00”

If a hierarchical value is to be assigned, it must be passed with all parent values where each level of the hierarchy is separated with "\". For example: Cars\BMW\X5.

}

Examples of data to assign:

- Assigning a value of the tag “Color Label”:

```
{
      “guid”: “f3dc7b01-24f3-49b4-93d2-5183c677984b”,
      “values”: [
            “Blue”
      ]
}
```

- Assigning a value of the tag “ License Expiration Date”:

```
{
      “guid”: “5872752c-ac94-406b-a025-770972ae2969”,
      “values”: [
            “2023.10.20.00.00.00”
      ]
}
```

- Assigning multiple values of the tag “Categories”:

```
{
      “guid”: “b855732d-7747-464b-9712-cc977c2f4da6”,
      “values”: [
            “Animals\\Cat”,
            “Places\\Home”
      ]
}
```

Getting Export Presets

Request

Method:

`GET`

Endpoint:

`/api/settings/getExportPresets`

Headers:

`Cookie: .AspNet.ApplicationCookie={cookie}`

Identification cookie received after the authentication

Example:

```
GET /api/settings/getExportPresets HTTP/1.1
Host: test.daminion.net
Cookie:
.AspNet.ApplicationCookie=aoHzhsQUMgPtZbm3tDixCHSEqK4ZxqvAY4Qg
ZX3jrqxvqXkjViLxpd39bk_FV4NRd6ViLoCQrQ3noa6YIjfTwKTIdCvFkTOPXf
YXXH8LY2hIMlJGaBIkoKs2Mk16H-_bYYDkmmtgfOj4F57KzaN_mQG0NXf-fJub
Edpgi9aQ29Vo5RC9KgRfOEBcrJbPq-nrwdmhVzVqX_5SiEkvsIzcoKA-AVAYCC
bWtYcGPOVqDztQDdXGJk-4Am1mxxEq6NqEVrmq8vv3EmvE6nAO04zLX3FBs0no
rabaxYJMOTLScXDBX2LhWqTWXOi_-rLhomGzY3o4zo7rcDJ227jFP7mHIotjUy
bNPQaoQKXb9__K24YQdRdgMWQi77KPF-QmUeNiPr3he0YZTRSEfG9c7cFnbeLT
lW-YjSOwPEuWNKZRTFE
```

Response

Headers:

`Content-Type: application/json; charset=utf-8`

Body: {

`“data”: **array(**ExportPreset**)**,`

Array of export preset.

`“error”: **string**,`

Error description.

"errorCode": **number**,

Error code.

"success": **boolean**

The status of the successful completion of the search.

}

ExportPreset: {

"guid": **string**,

Export Preset GUID.

"name": **string**,

Export Preset name.

"enabled": **boolean**,

If true, the export preset is enabled, otherwise it is disabled.

"format": **string**,

The format in which the original file will be exported.

"params": **Parameters**,

Additional export template options.

}

Preset: {

"converting": ConversionParameters,

Conversion parameters

"metadata": MetadataParameters,

Metadata parameters

"resizing": ResizingParameters,

Resizing parameters

"watermark": WatermarkParameters,

Watermark parameters.

}

ConversionParameters: {

"jpeG_Quality": **number**

The quality of the exported file in percent. The lower the quality, the higher the compression rate of the source image. This option is used only for JPEG templates.

}

MetadataParameters: {

"embeddingMethod": **number**

The method of embedding metadata during export. The following values are possible:

0 - data embedding is disabled

1 - embed all data

2 - embed all data except camera information

3 - embed only copyright information

}

ResizingParameters: {

"resizeUnit": **number,**

Unit for setting the size of the converted file. The following values are available:

0 - pixel

1 - inch

2 - centimeter

3 - percentage of the original file

4 - do not resize

"outputHeight": **number,**

Height of the output file

"outputWidth": **number,**

Width of the output file

"outputSizeInPercents": **number**

Size as a percentage of original.

}

WatermarkParameters: {

"watermarkGuid": **string**

The GUID of the watermark that is used in the export template.

}

Creating an embed link

Request

Method:

`POST`

Endpoint:

`/api/download/createCDNWithPreset`

Headers:

`Content-Type: application/json`

`Cookie: .AspNet.ApplicationCookie={cookie}`

Identification cookie received after the authentication.

Body: {

`"ids": array(number),`

The ID of the media item for which the embed link is being created. Note that the operation of creating the embed link is not a batch operation. If you specify multiple IDs in the parameter, the link will be created only for the media item with the first ID.

`"preset": string,`

The GUID of the export preset to use when creating the embed link. This is an optional parameter, if it is not specified, a link to the original file will be created.

}

Example:

```
POST /api/download/createCDNWithPreset HTTP/1.1
Host: test.daminion.net
Content-Type: application/json
Cookie:
.AspNet.ApplicationCookie=aoHzhsQUMgPtZbm3tDixCHSEqK4ZxqvAY4Qg
ZX3jrqxvqXkjViLxpd39bk_FV4NRd6ViLoCQrQ3noa6YIjfTwKTIdCvFkTOPXf
YXXH8LY2hIMlJGaBIkoKs2Mk16H-_bYYDkmmtgfOj4F57KzaN_mQG0NXf-fJub
Edpgi9aQ29Vo5RC9KgRfOEBcrJbPq-nrwdmhVzVqX_5SiEkvsIzcoKA-AVAYCC
bWtYcGPOVqDztQDdXGJk-4Am1mxxEq6NqEVrmq8vv3EmvE6nAO04zLX3FBs0no
rabaxYJMOTLScXDBX2LhWqTWXOi_-rLhomGzY3o4zo7rcDJ227jFP7mHIotjUy
```

```
bNPQaoQKXb9__K24YQdRdgMWQi77KPF-QmUeNiPr3he0YZTRSEfG9c7cFnbeLT
lW-YjSOwPEuWNKZRTFE
Content-Length: 154

{
    "ids": [360],
    “preset”: “1560d96e-a5b0-42a9-b6a3-92b08b64305b”
}
```

Response

Headers:

```
Content-Type: application/json; charset=utf-8
```

Body: {

“data”: **string**,

Relative path of the created embed link in the format: files/{prefix}/{hash}/{filename}, where hash is the unique hash of the link, prefix is the first 4 characters of the hash, filename is the original name of the file for which the link is created.

To get the absolute path, you need to add the URL of the catalog. For example, the catalog is sitting at https://test.daminion.net. After creating the link the relative path "files/86cy/86cydq86cydw/image.jpg" is obtained. The absolute path of the link should have the following form: "https://test.daminion.net/files/86cy/86cydq86cydw/image.jpg"

“error”: **string**,

Error description.

“errorCode”: **number**,

Error code.

“success”: **boolean**

Flag for the successful creation of an embed link.

}

Getting embed links

Request

Method:

`GET`

Endpoint:

`/api/download/getEmbedLinksById`

Query parameters:

`id`

The ID of the media item for which all embed links should be returned.

Headers:

`Cookie: .AspNet.ApplicationCookie={cookie}`

Identification cookie received after the authentication

Example:

```
GET /api/download/getEmbedLinksById?id=360 HTTP/1.1
Host: test.daminion.net
Cookie:
.AspNet.ApplicationCookie=aoHzhsQUMgPtZbm3tDixCHSEqK4ZxqvAY4Qg
ZX3jrqxvqXkjViLxpd39bk_FV4NRd6ViLoCQrQ3noa6YIjfTwKTIdCvFkTOPXf
YXXH8LY2hIMlJGaBIkoKs2Mk16H-_bYYDkmmtgfOj4F57KzaN_mQG0NXf-fJub
Edpgi9aQ29Vo5RC9KgRfOEBcrJbPq-nrwdmhVzVqX_5SiEkvsIzcoKA-AVAYCC
bWtYcGPOVqDztQDdXGJk-4Am1mxxEq6NqEVrmq8vv3EmvE6nAO04zLX3FBs0no
rabaxYJMOTLScXDBX2LhWqTWXOi_-rLhomGzY3o4zo7rcDJ227jFP7mHIotjUy
bNPQaoQKXb9__K24YQdRdgMWQi77KPF-QmUeNiPr3he0YZTRSEfG9c7cFnbeLT
lW-YjSOwPEuWNKZRTFE
```

Response

Headers:

`Content-Type: application/json; charset=utf-8`

Body: {

"data": **array(**EmbedLink**)**,

Array of embed links

"error": **string**,

Error description.

"errorCode": **number**,

Error code.

"success": **boolean**

The status of the successful completion of the search.

}

EmbedLink: {

"exportPreset": ExportPreset,

The export preset that was used to create the embed link. If this value is null, this link was created for the original file. The ExportPreset type is described in the Getting Export Presets section.

"fileHash": **string**,

Hash of the original file.

"url": **number**,

Relative path

}