

Sodyo

External Integration API

25 April 2018

Overview

Sodyo Server APIs enable developers to create/read/update/delete (CRUD) content and campaigns. Today users can do all of that via the UI, however, there are cases that a customer will want to automate the creation of content items, and campaigns via code, in order to tightly integrate it with internal workflow and systems.

Version 1 of the APIs will enable full campaign management, and full content management for Content of type “immediate action”. Content type of “Sodyo Ad” is not supported at this point.

Concept

- Super Admin is able enable/disable API support for project/account.
- Authentication
 - Account admin can configure API access keys to be used for integration
 - API authentication/authorization are based on access key
- Supported APIs
 - CRUD operations on content (limited to immediate action only)
 - Support content adaptation to strict API structure
 - CRUD operations on campaigns
 - Get marker image API, marker picture generation should be moved to server side
 - Campaign activation/deactivation and status
- Error handling

Authentication


Sodyo enable generation of app keys, that for security reasons, just the hash value of them is saved. Once you generate a new key, you need to copy it and use it in your code.

When the server get a request with a key name, he hash the value of the key and check it with the hash value that store in the server.

This way, no one can get the original key by penetrating the system.

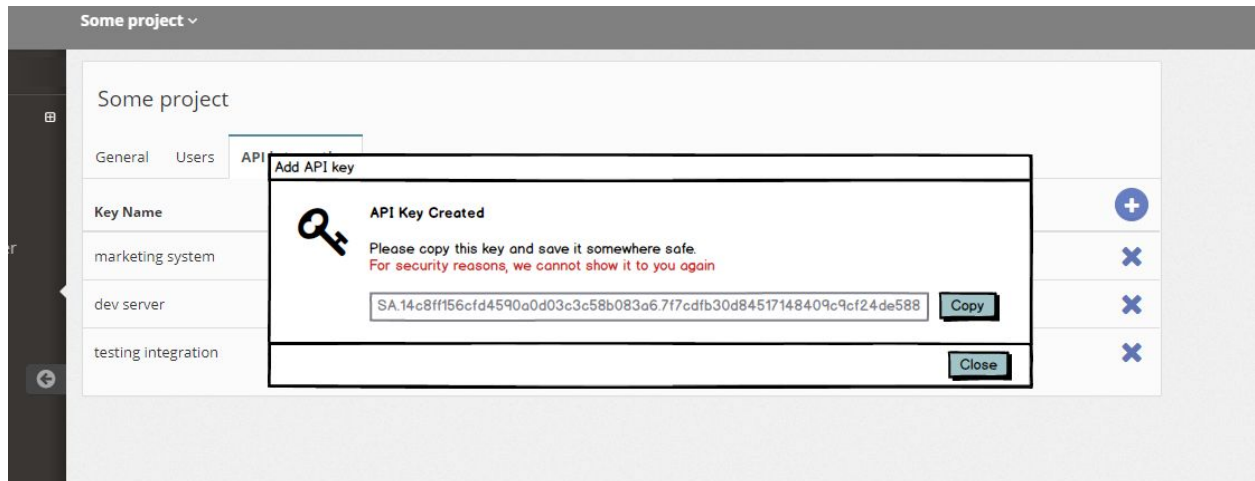
How does api keys work:

1. There is a tab in the project setting - API integration that is visible in case your project was open for Server API integration.
2. API section configuration is available to project admins and project owners, the same way as it is done for the rest of the settings
3. In the API integration section you will see a list of currently defined API keys for an account. Following columns are shown:
 - a. Key name
 - b. Key ID
 - c. Api key "placeholder". Hovering/clicking on key should show:



For security purposes, we do not allow the key value to be displayed.

- d. Delete key button.
4. Once deleted the key can not be used again for API access.
 5. Create new api provide a dialog with a single prompt - Key name
 6. CLicking on create and view will show the following screen:



7. API key have the following form:

SA.KEY_ID.HMAC_SIGNATURE

SA - sodyo API

For example:

SA.14c8ff156cfd4590a0d03c3c58b083a6.7f7cdfb30d84517148409c9cf24de588

8. Single API key can be used for single project only.

Authentication process:

1. Every API call have "X-AUTH-TOKEN" header with API key value.
2. Authentication is validated by the following way:
 - a. Validate API key exists
 - b. Validate HMAC signatureIf at least one check fails - a 401 Unauthorized message will be returned.
3. Every request authenticated with is authorized with new role API.
4. API role is eligible to access only external integration API endpoints in the system. Trying to access any other endpoint will return 403 Forbidden.
5. As a part of authorisation process it validate that API usage is available for a project. If API usage is disable for a project that the API key belongs to, the server will return 403 Forbidden.

API endpoint structure

1. The following endpoint pattern is used for all external integration API endpoints:

`/integration/rest/api/v1/entity/{entity-id}/action`

Where:

First part is static prefix for all

Second part is API version number

Third part is an entity name (campaign/content/...)

Fourth part is entity UUID

Fifth part is an action

Supported APIs

CRUD operations on content

1. All content operations available for immediate action content only.
2. If user tries to reference a SODYO_AD content 404 will be returned, the same as the content does not exists.

3. Required structure for content exposed via API:

```
"content": {
  "actionType": "PHONE", // Immediate action type
  "params": { // Map of immediate action properties according to
a type.
    "phone": "89787978978"
  }
}
```

API reference:

1. Ability to list all content available in account

HTTP GET /integration/rest/api/v1/content

Returns the list of the “IMMEDIATE_ACTION” content available on the account:

```
[
  {
    "uuid": "3d7f83f7-dcd5-4f03-ab97-2e1fb9056978",
    "name": "Call content",
    "description": "desc",
    "content": {
      "actionType": "PHONE",
      "params": {
        "phone": "89787978978"
      }
    }
  },
  {
    "uuid": "3d7f83f7-dcd5-4f03-ab97-2e1fb9056977",
    "name": "Data content",
    "description": "desc",
    "content": {
      "actionType": "DATA",
      "params": {
        "data": "89787978978"
      }
    }
  }
]
```

2. Ability to find content by name

HTTP GET /integration/rest/api/v1/content?name={content-name}

Example:

GET /integration/rest/api/v1/content?name=Call%20content

Response:

```
[
  {
    "uuid": "3d7f83f7-dcd5-4f03-ab97-2e1fb9056978",
    "name": "Call content",
    "description": "desc",
```

```
[
  {
    "content": {
      "actionType": "PHONE",
      "params": {
        "phone": "89787978978"
      }
    }
  }
]
```

3. Ability to get single content available in account

HTTP GET /integration/rest/api/v1/content/{UUID}

Returns single content entity.

Example:

GET /integration/rest/api/v1/content/3d7f83f7-dcd5-4f03-ab97-2e1fb9056977

Response:

```
{
  "uuid": "3d7f83f7-dcd5-4f03-ab97-2e1fb9056977",
  "name": "Call content",
  "description": "desc",
  "content": {
    "actionType": "DATA",
    "params": {
      "data": "89787978978"
    }
  }
}
```

4. Ability to delete content entity

HTTP DELETE /integration/rest/api/v1/content/{UUID}

Example:

/integration/rest/api/v1/content/3d7f83f7-dcd5-4f03-ab97-2e1fb9056977

Response:
HTTP 200 OK

5. Ability to update content entity

HTTP POST /integration/rest/api/v1/content/{UUID}

Example:

Request:

/integration/rest/api/v1/content/3d7f83f7-dcd5-4f03-ab97-2e1fb9056977

Body:

```
{
  "name": "Call content updated",
  "description": "desc",
  "content": {
    "actionType": "DATA",
    "params": {
      "data": "blah blah"
    }
  }
}
```

response :

```
{
  "uuid": "3d7f83f7-dcd5-4f03-ab97-2e1fb9056977",
  "name": "Call content updated",
  "description": "desc",
  "content": {
    "actionType": "DATA",
    "params": {
      "data": "blah blah"
    }
  }
}
```

6. Ability to create content entity

HTTP POST /integration/rest/api/v1/content

Example:

Request:

/integration/rest/api/v1/entity

Body:

```
{
  "name": "Call content updated",
  "description": "desc",
  "content": {
    "actionType": "DATA",
    "params": {
      "data": "blah blah"
    }
  }
}
```

response :

```
{
  "uuid": "3d7f83f7-dcd5-4f03-ab97-2e1fb9056977",
  "name": "Call content updated",
  "description": "desc",
  "content": {
    "actionType": "DATA",
    "params": {
      "data": "blah blah"
    }
  }
}
```


CRUD operations on campaigns

1. The API work only with campaigns associated with “immediate action” content
2. API mask the campaign group by seamlessly providing a default campaign group when creating/updating a campaign.

API reference:

1. Get a list of all available campaigns in the account:

HTTP GET /integration/rest/api/v1/campaign

Example:

Request:

GET /integration/rest/api/v1/campaign

Response:

```
[
  {
    "uuid": "41c700f3-754c-42fe-a64b-141865829dee",
    "name": "monitoring campaign",
    "mediaType": "TV",
    "location": "Mobile",
    "notes": null,
    "address": null,
    "contentUuid": "85971fe1-4236-4ee0-b292-2f648c72d519",
    "contentName": "monitoring content",
    "createDate": "2017-06-27T12:09:20.523+0000",
    "updateDate": "2018-04-11T13:38:21.666+0000",
    "enabled": true
  },
  {
    "uuid": "bdbfff6a-4eac-4476-9f9c-bb5e9382727d",
    "name": "some campaign",
    "mediaType": "Billboard",
    "location": "Stationary",
    "notes": "some notes",
    "address": "some address",
```

```
"contentUuid": "3d7f83f7-dcd5-4f03-ab97-2e1fb9056977",  
"contentName": "Call content",  
"createDate": "2018-04-16T08:07:47.569+0000",  
"updateDate": "2018-04-16T08:07:47.569+0000",  
"enabled": true  
}  
]
```

2. Find campaign by name

HTTP GET /integration/rest/api/v1/campaign?name={name}

Example:

Request:

GET /integration/rest/api/v1/campaign?name=monitoring%20campaign

Response:

```
[  
  {  
    "uuid": "41c700f3-754c-42fe-a64b-141865829dee",  
    "name": "monitoring campaign",  
    "mediaType": "TV",  
    "location": "Mobile",  
    "notes": null,  
    "address": null,  
    "contentUuid": "85971fe1-4236-4ee0-b292-2f648c72d519",  
    "contentName": "monitoring content",  
    "createDate": "2017-06-27T12:09:20.523+0000",  
    "updateDate": "2018-04-11T13:38:21.666+0000",  
    "enabled": true  
  }  
]
```

3. Get single campaign:

HTTP GET /integration/rest/api/v1/campaign/{uuid}

Example:

Request:

GET /integration/rest/api/v1/campaign/41c700f3-754c-42fe-a64b-141865829dee

Response:

```
{
  "uuid": "41c700f3-754c-42fe-a64b-141865829dee",
  "name": "monitoring campaign",
  "mediaType": "TV",
  "location": "Mobile",
  "notes": null,
  "address": null,
  "contentUuid": "85971fe1-4236-4ee0-b292-2f648c72d519",
  "contentName": "monitoring content",
  "createDate": "2017-06-27T12:09:20.523+0000",
  "updateDate": "2018-04-11T13:38:21.666+0000",
  "enabled": true
}
```

4. Create campaign

HTTP POST /integration/rest/api/v1/campaign

Example:

Request:

POST /integration/rest/api/v1/campaign

Request body:

```
{
  "name": "monitoring campaign",
  "mediaType": "TV",
  "location": "Mobile",
  "notes": null,
  "address": null,
  "contentUuid": "85971fe1-4236-4ee0-b292-2f648c72d519",
  "enabled": true
}
```

Response:

```
{
  "uuid": "41c700f3-754c-42fe-a64b-141865829dee",
  "name": "monitoring campaign",
  "mediaType": "TV",
  "location": "Mobile",
  "notes": null,
  "address": null,
  "contentUuid": "85971fe1-4236-4ee0-b292-2f648c72d519",
  "contentName": "monitoring content",
  "createDate": "2017-06-27T12:09:20.523+0000",
  "updateDate": "2018-04-11T13:38:21.666+0000",
  "enabled": true
}
```

5. Update existing campaign

HTTP POST /integration/rest/api/v1/campaign/{uuid}

Example:

Request:

POST /integration/rest/api/v1/campaign/41c700f3-754c-42fe-a64b-141865829dee

Request body:

```
{
  "uuid": "41c700f3-754c-42fe-a64b-141865829dee",
  "name": "updated monitoring campaign",
  "mediaType": "TV",
  "location": "Mobile",
  "notes": null,
  "address": null,
  "contentUuid": "85971fe1-4236-4ee0-b292-2f648c72d519",
  "createDate": "2017-06-27T12:09:20.523+0000",
  "updateDate": "2018-04-11T13:38:21.666+0000",
  "enabled": true
}
```

Response:

```
{
  "uuid": "41c700f3-754c-42fe-a64b-141865829dee",
  "name": "updated monitoring campaign",
  "mediaType": "TV",
  "location": "Mobile",
  "notes": null,
  "address": null,
  "contentUuid": "85971fe1-4236-4ee0-b292-2f648c72d519",
  "contentName": "monitoring content",
  "createDate": "2017-06-27T12:09:20.523+0000",
  "updateDate": "2018-04-11T13:38:21.666+0000",
  "enabled": true
}
```

Get marker image API

1. The api support providing a required marker image dimensions, and return the image fitting the provided parameters.
2. Image is in a PNG format
3. Image have rounded corners like it is works in the CMS portal
4. Image have transparent background
5. Supported sizes are limited on both x and y to [24:2400] pixels



API reference:

1. Get campaign marker:

HTTP GET /integration/rest/api/v1/campaign/{uuid}/marker?x=300&y=900

Response:

marker image binary stream



Campaign activation/deactivation and status

1. All campaigns are enabled by default, however the API should allow enabling/disabling specific campaigns

API reference:

1. Enable campaign marker:

HTTP POST /integration/rest/api/v1/campaign/{uuid}/enabled/true

2. Disable campaign marker:

HTTP POST /integration/rest/api/v1/campaign/{uuid}/enabled/false

Error handling

1. Trying to access the entities which don't belong to a API key account, will return HTTP 404 - not found
2. Sending non valid entity will return the HTTP 400 -bad request with validation error
3. Trying to access the non existent entities will return HTTP 404 - not found
4. Trying to access the API without a valid authorization header will return HTTP 401 Unauthorized