ALMANAPP®

API MANUAL

REF: AL-EAM-001

Releases

Version	Authors
0.1	Hielke Fellinger
0.5	Hielke Fellinger
0.6	Hielke Fellinger
0.9	Hielke Fellinger
0.9.1	Hielke Fellinger
0.9.2	Hielke Fellinger
0.9.3	Hielke Fellinger
0.9.4	Bas Krijgsman
0.9.5	Bas Krijgsman

Changes

Version	Comments
0.1	Document Initialisation
0.5	Document Alfa release
0.6	Content updates
0.9	Document Beta release
0.9.1	Call updates
0.9.2	Added Authentication check call
0.9.3	Nullable FK columns now have default of NULL. Field length increased varchar to minimum of 128 on most title and/or name fields.
0.9.4	Added Authentication creation
0.9.5	Added Authentication search call

V 0.9.5 2016-11-14

Table of Content

1. Introduction

This manual covers the information needed to interact with the first version of the external API. The external API focuses on enabling basic interaction with the data behind Almanapp software like the Almanapp APPMIN and the Almanapp APP itself. This manual presupposes a working knowledge of an API's workings, basic programming skills and a knowledge of the accompanying literature and terms.

1.1 Philosophy

The main use of this first version is to check the need for an API to enable basic resource synchronization to and from others sources. This version will be supplied (mostly) inline with REST and OAuth. Further development will be depending on the size of the userbase and the set of features that can be enabled for current and potential customers. Possible further developments will most likely include 3rd party access to supply their own custom, features on the Almanapp APP Platform at the desire of customers.

This version of the API (v1) will remain online until, at the very least, the 31th December 2016 on the base of best effort (implicit and explicit guarantees can not be supplied). Changes and Fixes to the API will be reflected in updated versions of this document as soon as possible. Alterations to calls and their payloads itself will be avoided and will, most likely, result in an available call on a higher version number.

1.2 General Technology

In the current version of the API responses will always be send as JSON (UTF 8; IETF RFC4627) as a response to a HTTP request, however the groundwork for XML responses are build in the background so this return type might become available in the future.

2. Version One /v1/

This chapter is to explain the interaction you can have with the /v1/ calls available in the external API. Furthermore it will list the calls and possible responds those calls will deliver.

2.1 Call Basics

The /v1/ calls respond only to the HTTP Request types GET and POST, these are used for the entire CRUD set. The HTTP request types PUT and DELETE will be responded with an error. The correct request type and call are listed later on in this chapter.

In the aid of readability most Url's have been shortened in this manual as following:
'https://{BASE}/{UNION}/api/v1/{RESOURCE}/attribute1/attribute2' to
'https://{URL}/{RESOURCE}/attribute1/attribute2'

Call Structure:
All the calls elements use lowercase, an exception on this rule may be the attributes (6..n)

C2.EX1 (UKN.)	**https://almanapp.nl/union/api/v1/news/attribute1(/attribute2(/))**

Order	Call url part	Comments
0	http(s)://	HTTP and HTTPS available, HTTPS recommended and possible required in the future. In all examples HTTPS is used
1	(www.)almanapp.nl/	Base of URL *further known as: {BASE}*
2	union/	The Union or Company name you want CRUD access to *further known as: {UNION}*
3	api/	Select the API
4	v1/	The version selector of the call, in this case: "v1"
5	news/	The Resource (Always Singular: "user instead of users") you want CRUD access to *further known as: {RESOURCE}*
6..n	attribute1(/attribute 2(/))	The attributed or filters to a call. The complexity is mostly moved to the parameters of the corresponding HTTP request

2.2 Response Basics

All responses types will be delivered encapsulated in a JSON Object (0...1 Items) or JSON Array in the case in the possibility of returning, multiple values (0...N). In the current version a limit attribute (to limit the size of the total return) is not yet available.

In this version there are two basic response type; The error response and the success response. In both cases JSON will be returned solang as the requests are made inside of the API's reach. More about this subject in Chapter 2.2.2

2.2.1 Header Structure

Name	Payload
Cache-Control	no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Connection	Keep-Alive
Content-Encoding	gzip
Content-Type	{VARIABLE}
Date	{CURRENT_DATE}
Expires	Thu, 19 Nov 1981 08:52:00 GMT
Keep-Alive	timeout=1, max=100
Pragma	no-cache
Server	Apache/{VERSION}
Vary	Accept-Encoding,User-Agent
X-Powered-By	PHP/{VERSION}

The table above is the header returned from a HTTPS request. The current response header focuses on stopping and eliminating request buffering, ensuring a more up-to-date result as long as the header is not ignored.

2.2.2 Response Structures

Like stated in the beginning of Chapter 2, There are only two basic response types; the error and the success response. Both of the response types will result in either a JSON message as a result. Any status code other than the HTTP 200: “OK” will result in the return of an error response.

Depth	Content	Comments
0	[...] / {...}	JSON Array / Object
1	Attribute and there values variables in case of a single object or zero to multiple JSON Objects containing their values in the case of an JSON Array in depth zero.	None

The success response comes in two different formats: as a JSON Array or Json Object. The default return type is always a JSON Object. Note: A JSON Array is only and always used when the call has the potential to return multiple items, even if the result set is smaller than two to keep consistency.

```
{
   "error": {MESSAGE}
}
```

The error response will always be encapsulated in a JSON Object; according to the image above. The error variable payload ({MESSAGE}) will be in English by default. Later this year more languages will be supported. An exception to the rules stated above is in the case of a total failure of the API Facade, in this case no guarantees can be given.

Used HTTP Status response codes:

Code	Description
200	All successful responses will be supplied with an 200 OK. Response codes 201 to 2xx are not used in v1 calls
400	Bad request or bad request payload syntax. Used only in Create, Update, Batch and Delete Calls
403	Authentication failed (Token request). Faulty or no token supplied in other calls
404	Call address with an specific HTTP request or the requested resource(s) have not been found.
500	Internal Server error

2.3 Authentication & Security

In this Chapter the basic authentication and security features are explained. A more comprehensive security description, of the items behind the API facade, are not included in this document.

2.3.1 Authentication

Authentication in the API is done via is done via a token, This token can be obtained via a successful HTTP GET request to the call: “C2.01A”. This token is required in every call, except the handshake itself. The result of the query are handled later on in this chapter (Token Request Result)

C2.01A (GET)	{URL}/handshake/{username}/{password}(/{description})

Resource	Description	Optional
{username}	A username name of an ‘admin’ or ‘superadmin’ of the association given in {UNION}	No
{password}	The password matching to the account given in {username}. The value of the password variable is removed before request logging occurs.	No
{description}	Short, single word, description for the base of creating a token	Yes

Token Use
Recommended is, if an automated task is not run daily or at a smaller interval, to create a new token for every request to be ensured of access. To facilitate this, an ‘admin’ (default management account) or ‘superadmin’ user can create up to 64 active tokens.

To use a token in a call, where authentication is required, append the following (HTTP GET parameter) to the base of the URL: “?token={TOKEN}” (Example: C2.EX2). If you use the token where no authentication is needed, the token is ignored and not eligible to the automatic extension of its expire date as described in previous alinia.

C2.EX2 (UKN.)	{URL}/{RESOURCE}/{attribute1}(/)?token={TOKEN}

Token Request Result
The result of a successful token request call (C2.01A) is listed below.

```
{
   "token":  {TOKEN},
   "expire": "2015-12-31 23:59:59",
   "description": {description},
   "securityLevel": "0"
}
```

Attribute	Description
token	A 64 character long randomized string consisting of lowercase alphanumeric character.
expire	The expire date of the token, default is set to seven days from the date of the initial request. If the token is used in within the last 24 hours of validity, the expire date in increased with 48 hours
description	Short, single word, description of why this token was created. This attribute is optional, if no description has been supplied an empty string is returned.
securityLevel	In this version only level 0 (zero) is available, indicating access to all calls mentioned in Chapter 2

2.3.2 Checking user credentials

For the purpose of remote authentication a new call is introduced since 2016-02-29. This call allows the checking of user credentials remotely for API users with a token (2.3.1 Authentication).

C2.01B (GET)	{URL}/check/credentials/{username}/{password}

Checking user credentials this way does not grant access of that user to the API or any other software in the Almanapp ecosystem. This call is purely used to verify the validity of the supplied credentials. Users who are disabled will always result in the return of false.

Credential Checking Result
The result of a successful request call (C2.01B) is listed below.

```
{
   "result":  {true/false},
}
```

2.3.3 Security

The term security in this manual and in this chapter cover only the data relevant to a user of the API in a really abstract and broad form. Security Guidelines, Concepts and Implementation thereof behind and inside the API facade are not covered in this manual at all.

Security in the API focuses itself broadly on three key attributes: Confidentiality, Integrity and Availability (CIA Triad). Talking on a superficial level, the confidentiality attribute is covered by the token authorization (see Chapter 2.3.1) and the requirement of an ‘admin’ or ‘superadmin’ account for the creation of a token in the first place. However there no option to create a token for a specific operation, call or resource. All tokens are created with equal rights (Possibility to execute all calls). Sharing your token with a or multiple 3rd parties and/or sending request through HTTP and not through HTTPS is extremely discurished.

Another important security attribute is the integrity of the data in the database. CRUD Calls to the API are checked on the properties (and their limits) of the resources expected by the database but nothing more. However there is an option to work with Transactions, via bash calls (See Chapter 2.3.5), to enable conditional updates and possible item creation. On the base of integrity please take note that the users have the possibility (by default) to edit their own details in the App, making the Almanapp Database content possibly more up-to-date than your current (local) user data.Last but not least is the security attribute Availability. A general description of the security attribute to provide (timely) access to resources. Like stated in Chapter 1.2 this API, gives no guarantees on (and not limited to) the metrics like Availability and Performance. That said it is an upcoming focus.

2.4 Basic Actions (CRUD)

This Chapter covers the basic/default CRUD actions and the Batch update features of the API almost every resource available from the API. The following resources, used as implementation generic placeholder {RESOURCE}, are available for the mentioned CRUD actions in this chapter (2.3):

Resource	Description
calendar	The calendar resource is used to display dates and their details for agenda items.
group	Defined here as a collection of users, it might be an organisational unit, location or social club within the union or company.
groupcategory	The types of groups there are inside the union. Types could be like stated above; an organisational unit, location or social club.
news	News items and their details.
pages	General pages, not stuff like news, events or agenda items, Stuff like The history of a union or the
user	The members and or employees of the union, all of them including users with the ‘admin’ and ‘superadmin’
event	Scheduled events of the union
poll	Contains the base data of the polls
galleries	Contains the photo albums of the Union
galleryimage	Contains Album images

Details on the resources and these variables are found in the Apendix: A. At the moment all CRUD calls are done through the POST and GET HTTP request. The DELETE request, for deletion, and the PUT request, for updates (W3C RFC2616), are both replaced by an url alteration coupled with a POST request.

There are some secondary order attributes who differ slightly for there first order cousins. The changes form the first order cousins are mainly on the requirement of the id attribute of there first order cousens. The HTTP response codes noted in the chapters 2.4.x will remain the same for the second order resources

Resource 2nd Order	Description
poll/vote	The passed votes on the different polls
poll/options	The option of the different polls
event/lineup	The possible lineup of the an event

The following CRUD calls will differ slightly for the second order attribute:

Create

C2.02 2nd (POST)	{URL}/{RESOURCE 1ST}/{id RESOURCE 1ST}/{RESOURCE 2ND}/

So the create calls of the 2nd order resource vote on the 1ste poll might look something like this:

C2.02 EX (POST)	{URL}/poll/1/votes/

Read

The following calls are to receive all second order resources and a second order resource bij its id respectably.

C2.03 2nd (GET)	{URL}/{RESOURCE 1ST}}/{id RESOURCE 1ST}/{RESOURCE 2ND}/

C2.04 2nd (GET)	{URL}/{RESOURCE 1ST}/}/{id RESOURCE 1ST}/{RESOURCE 2ND}/ {id RESOURCE 2ND}

2.4.1 Create

C2.02 (POST)	{URL}/{RESOURCE}

The creation of a new resource is done via a call POST request to the root of the resource. The POST collection acts as an key value storage of the object you want to create. The values received will undergo explicit type conversion. Please take note that if keys are found that don't match with the attributes of the object they are ignored without feedback.

If the insertion of the new object is a success (HTTP status 200: "OK") a success response will be send containing the newly created object for client side verification. The returned resource object is not subject to the explicit type conversion, if you send an attribute as String value, containing the for that object's attribute 'required' int, the API will type conversion to the database but not to the returned object.

Used HTTP Status response codes:

Code	Description
200	Everything went successful. The newly created object of the resource is returned.
400	Object is not created due to bad syntax and/or missing
500	Internal Server error

2.4.2 Read

C2.03 (GET)	{URL}/{RESOURCE}

C2.04 (GET)	{URL}/{RESOURCE}/{id}

There are multiple ways of getting instances of the resources. The default ways are the get by ID (C2.03) and the get entire collection (C2.03) calls. At the moment there is no option to enact a Limit on the number of returned objects (Call C2.03 & C2.05), so be aware of possible large waiting times and high data throughput for call result of large collections.

C2.05 (POST)	{URL}/{RESOURCE}/search(/{operator}(/{like}/(negate)))

Resource	Description	Optional
{operator}	The operator used in the underlying SQL type statement ‘and’ or ‘or’. Every string except ‘or’ will result in the use of ‘and’. *Default: ‘and’, not case sensitive.*	Yes
{like}	Cast all string types in a SQL LIKE ‘%{VALUE}%’ type construction for the underlying SQL type statement. Every string except ‘false’ and ‘no’ will result in the use of ‘true’. *Default: ‘false’ not case sensitive.*	Yes
{negate}	Negate the entire selection form: “select all that” to: “select all that do not have”. Every string except ‘false’ and ‘no’ will result in the use of ‘true’. *Default: ‘false’, not case sensitive.*	Yes

As in most REST implementation filtering of resources is usually done by extending the url and/or adding multiple GET parameters, however this is not yet supported in the current version. An alternative is delivered in the form of the search call (C2.05). This call enables filtering through the use of the POST attribute as a key values store (The values received will undergo explicit type conversion). Just like the create an update statement the POST attributes (key value) are used to represent the attributes and their value. For the moment is it only possible to use one value per attributes for the filtering.

Used HTTP Status response codes:

Code	Description
200	Everything went successful. The newly created object of the resource is returned.
400	Object is not created due to bad syntax and/or missing variables
404	Object is not found (Applicable to C2.03 & C2.04)
500	Internal Server error

2.4.3 Update

C2.06 (POST)	{URL}/{RESOURCE}/{id}

Resource	Description	Optional
{id}	The API Id of the resource object you would like to change.	No

Instead of a PUT request this call is handled by an POST request following the same rules as the creation of an object of a resource (chapter 2.3.1). It might be a hassle to update multiple items, for that functionality the batch call has been created see: Chapter 2.3.3

Used HTTP Status response codes:

Code	Description
200	Everything went successful. The newly updated object of the resource is returned.
400	Object is not updated due to bad syntax and/or missing variables
404	The Object you want to update can not be found
500	Internal Server error

2.3.4 Delete

C2.07 (POST)	{URL}/{RESOURCE}/delete/{id}

Resource	Description	Optional
{id}	The API Id of the resource object you would like to delete.	No

Instead of a DELETE request this call is handled by a POST request to delete a resource object. Further values and the handling of the POST attributes are not done. WARNING: All linked data to this object and this object itself are deleted and not just disabled.

Used HTTP Status response codes:

Code	Description
200	Everything went successful. The just deleted object of the resource is returned.
400	Object is not updated due to bad syntax and/or missing variables
404	The Object you want to update can not be found
500	Internal Server error

2.4.5 Batch

C2.08 (POST)	{URL}/batch/{RESOURCE}

The batch call is a combination of the create and the update call for multiple objects (1...n) of the resource. As an extra feature the batch procedure runs in a SQL type Transaction; meaning it will 'reserve' the changes in the database and if no faults have been found the changes will be committed else an error message of the first found error is returned.

POST attribute	Description
json	Containing all the changes in a big JSON Array, following the resource attribute requirements and naming. Appendix: A
uniqueKeys	The set of resource attributes that create a object identifier (Primary Key). Example: The combination of 'name' and 'birthday' make a unique cobination for all objects you would like to commit in the JSON Array: So for this example the value of uniqueKeys will be: ["name","birthday"]

The way to handle this call is rather different form the other calls; it uses the POST request in a different way (as seen in the table above). This call is designed for high numbers of data, and only returns the first error it encounters and not all the errors of the whole set inside of an error response (See an example below).

```
1  {
2     "error": "Entity: {i} of {n} Has errors during updating  Dump: {"error":["Set Field does
..  not exist: {wrong value}"]}"
3  }
```

Like stated earlier, the batch call is a combination of the create and the update call for multiple objects (1...n). This call parses every send resource object (in the POST json attribute) one by one and if the provided uniqueKeys do not supply a match within the database the resource object is entered as a new database element. If one element has been found it overwrites the object in the database with the supplied resource object,

however if there are more than 1 objects found as result of the uniqueKeys combinations an error will be provided.

At the moment it is not able to change the batch call settings (like to only update, create or delete) Please take note that the limits of this call have not been fully tested, however do to the use of transactions it does not commit the changes until you receive a HTTP 200: “OK” response.

Used HTTP Status response codes:

Code	Description
200	Everything went successful. returns a JSON Object containing a scalar with the value true
400	The entire set of Objects (1..n) is not updated or created due to bad syntax and/or missing variables.
500	Internal Server error

2.5 Extra Calls

This Chapter covers the extra calls available to some of the Resources, these extra calls are grouped by resource in the following sub-chapters. These extra calls are mainly there to supply connections/links between the main resources (‘Junction Tables’ in SQL terminology).

2.5.1 User

The user resource is in widely used in connection of multiple resources at the moment only the interaction between the user resource and the group resource is enabled. Only Create and Read calls are available.

C2.09 (GET)	{URL}/user/{id}/group

Resource	Description	Optional
{id}	The API Id of the user object you would like to see all the groups and the functions within those groups from.	No

At the moment there is no option to enact a Limit on the number of returned objects, so be aware of possible large waiting times and high data throughput for call result of large collections on slow connections.

Used HTTP Status response codes:

Code	Description
200	Everything went successful. Return a JSON Array containing functions
400	Object is not created due to bad syntax and/or missing variables
500	Internal Server error

C2.10 (POST)	{URL}/user/{id}/group/{groupId}(/{function})

Resource	Description	Optional
{id}	The API Id of the user object you would like to alter.	No
{groupId}	The API Id of the group object you would like to add to the user.	No
{function}	The possible function of the user in the group	Yes

Just the request is is needed to complete the request, no payload in the POST request and actively ignored. If the insertion of the new relation is a success (HTTP status 200: “OK”) a success response will be send containing the newly created object for client side verification. The returned resource object is not subject to the explicit type conversion, if you send an attribute as String value, containing the for that object’s attribute ‘required’ int, the API will type conversion to the database but not to the returned object.

Further calls Update and Deletion of users to a group will come available soon, to completely enable CRUD all the core Almanapp APP data.

Used HTTP Status response codes:

Code	Description
200	Everything went successful. Return a JSON Array containing
404	The Object you want to update can not be found
500	Internal Server error

C2.11 (GET)	{URL}/user/{id}/auth/id

Resource	Description	Optional
{id}	The API Id of the user object you would like to alter.	No

This call enables the translation of a User id to a auth Id. This call is created to support relations on other tables via a requested auth id.

Used HTTP Status response codes:

Code	Description
200	Everything went successful. Return a JSON Array containing authId
404	The Object you want to update can not be found
500	Internal Server error

APPENDIX: A: /v1/ resources

Resource	Description
calendar	The calendar resource is used to display dates and their details for agenda items.
calendargroup	Categories of calendar available
group	Defined here as a collection of users, it might be an organisational unit, location or social club within the union or company.
groupcategory	The types of groups there are inside the union. Types could be like stated above; an organisational unit, location or social club.
news	News items and their details.
pages	General pages, not stuff like news, events or agenda items, Stuff like The history of a union or the
user	The members and or employees of the union, all of them including users with the ‘admin’ and ‘superadmin’
poll	Contains base data of the polls
event	Contains the events scheduled
eventcalendar	Links an event and calendar item together. Also used to track event participation and subscription.
gallery	Photo albums of the union
galleryimage	Photo’s inside the album

Resource 2nd Order	Description
poll/vote	The passed votes on the different polls
poll/options	The option of the different polls
event/image	The possible Images of the an event
event/speaker	The possible Speakers of the an event
eventcalendar/lineup	The possible lineup of the an eventcalendar
eventcalendar/speaker	The possible Speakers of the an eventcalendar
eventcalendar/participant	The Participants to an eventcalendar

All Resources above have been supplied with an ‘id’ attribute as follows:

Attribute	Type	Optional	Description
id	int(11)	No	Unique identifier in the API. When updating it will be created (than it is optional).

The ‘id’ attribute serves as the primary key for all the resources mentioned above. When the manual references ‘API id of a {RESOURCE} object’ this is the value that is mentioned

A. Calendar

Values and requirements

Attribute	Type	Optional	Description
title	varchar(64)	No	Event Title
image	varchar(128)	No	URL, Image will be buffered in local image server. Empty string allowed
text	mediumtext	No	Text about the event. Empty string and HTML allowed.
json	mediumtext	No	Extra attributes, created by Almanapp APPMIN. Empty string allowed
start	timestamp	No	Date in Unix timestamp
end	timestamp	No	Date in Unix timestamp
created_on	timestamp	No	Date in Unix timestamp
created_by	timestamp	No	Date in Unix timestamp
rights	int(11)	Yes	Who will be able to view the data. 0 (zero) no one, 1 (one) Guests, 2 (two) Users. Works as binary
public	tinyint(1)	No	Who will be able to view the data. 0 (zero) Users, 1 (one) The rest.
event_ calendar_id	int(11)	Yes	Links to API ID of event calendar (not yet supported)
calendar_ group	int(11)	No	Links to API ID of a calendargroup object. Allowed to be 0 (zero)

A. Calendargroup

Values and requirements

Attribute	Type	Optional	Description
name	varchar(128)	No	Group Name
color	varchar(7)	No	RGB color in hex. Like #DDDDDD

A. Group

Values and requirements

Attribute	Type	Optional	Description
name	varchar(128)	No	Group Name
image	varchar(128)	No	URL, Image will be buffered in local image server. Empty string allowed
year	int(11)	No	Year of creation
public	tinyint(1)	No	Who will be able to view the data. 0 (zero) Users, 1 (one) The rest.
category	timestamp	No	API id of a groupcategory object
data	mediumtext	Yes	(Legacy)
data_extra	mediumtext	Yes	(Legacy)
external_id	varchar(254)	Yes	External Identifier
date_updated	timestamp	Yes	Last time updated

A. Auth

Values and requirements

Attribute	Type	Optional	Description
name	varchar(128)	No	Login name
user_id	int(11)	No	Id of the user
right_group	enum(10)	No	‘user’ or ‘admin’
disabled	tinyint(1)	No	0 = Can login 1 = Can’t login
external_id	varchar(254)	Yes	External Identifier
date_updated	timestamp	Yes	Last time updated

Batch calls are not available on this route.

Attributed only available on create or update:

Attribute	Type	Optional	Description
password	varchar(128)	Yes	Password (Plain)

A. Groupcategory

Values and requirements

Attribute	Type	Optional	Description
name	varchar(128)	No	Group Category Name
color	varchar(10)	No	RGB color in hex. Like #DDDDDD
icon	varchar(256)	No	URL, Image will be buffered in local image server. Empty string allowed
category_group_by	varchar(32)	Yes	Set sorting in list year or name
user_list_group_by	varchar(32)	Yes	Set sorting in list year or name
external_id	varchar(254)	Yes	External Identifier
date_updated	timestamp	Yes	Last time updated
type	varchar(256)	Yes	Type Description

A. News

Values and requirements. The resource will require some work. The addition of a Auth (Authentication) object or the retrieval of Auth information via the user calls

Attribute	Type	Optional	Description
user	int(11)	No	API id of a auth object, will be set to a default
public	tinyint(1)	Yes	(Legacy) Who will be able to view the data. 0 (zero) Users, 1 (one) The rest.
title	varchar(128)	No	News Title
image	varchar(128)	No	URL, Image will be buffered in local image server. Empty string allowed
text	mediumtext	No	Text about the news. Empty string and HTML allowed.
created_on	timestamp	No	Date in Unix timestamp
attachments	json	No	Special attachments holder. Empty string and HTML allowed .
external_id	varchar(254)	Yes	External Identifier
date_updated	timestamp	Yes	Last time updated
flags	int(11)	Yes	Default 0 (zero) Not yet implemented
time	int(11)	No	Timestamp of launch
link	varchar(1024)	No	Linked item as URL

A. Page

Values and requirements. The resource will require some work. The addition of a Auth (Authentication) object or the retrieval of Auth information via the user calls

Attribute	Type	Optional	Description
auth_id	int(11)	No	API id of a auth object, will be set to a default
image	varchar(128)	No	URL, Image will be buffered in local image server. Empty string allowed
public	tinyint(1)	Yes	(Legacy) Who will be able to view the data. 0 (zero) Users, 1 (one) The rest.
created_on	timestamp	No	Date in Unix timestamp (Creation)
title	varchar(64)	No	Page Title
text	longtext	No	Text about the news. Empty string and HTML allowed.
attachments	json	No	Special attachments holder. Empty string and HTML allowed .
external_id	varchar(254)	Yes	External Identifier
date_updated	timestamp	Yes	Last time updated
link	varchar(1024)	No	Linked News or other item as URL

A. User

Values and requirements. Check current use of field before use.

Attribute	Type	Optional	Description
identifier	varchar(128)	Yes	External Unique identifier (PK)
second name	varchar(32)	Yes	..
middlename	varchar(32)	Yes	..
firstname	varchar(32)	Yes	..
bank_ number	varchar(42)	Yes	..
bank_user_ field	varchar(32)	Yes	Name of account owner used for sending. Example ‘G. Orwell’
phone1	varchar(64)	Yes	Landline of user
phone2	varchar(32)	Yes	Mobile Phone of user
phone3	varchar(32)	Yes	..
email	varchar(32)	Yes	User’s email
email2	varchar(128)	Yes	..
gender	varchar(1)	Yes	Gender M/F (Or local language)
year	int(11)	Yes	Year formated as ‘2001’
birthday	varchar(32)	Yes	Date of birth
image	varchar(128)	Yes	URL, Image will be buffered in local image server. Empty string allowed
search_name	varchar(32)	Yes	Name used for searching
study_name	varchar(32)	Yes	Study subject
student_nr	varchar(32)	Yes	Student id/number
contact_name	varchar(32)	Yes	Contact (like parents etc.) name

Continues on next page.

A. User (cont.)

Values and requirements. Check current use of field before use.

Attribute	Type	Optional	Description
join_date	varchar(32)	Yes	...
educational_direction	varchar(3)	Yes	Field of study
faculty	varchar(32)	Yes	..
study	varchar(3)	Yes	Name of study
lastname	varchar(64)	Yes	..
initials	varchar(10)	Yes	..
title	varchar(10)	Yes	..
address1_city	varchar(32)	Yes	First address
address1_country	varchar(32)	Yes	..
address1_number	varchar(32)	Yes	..
address1_number_ addition	varchar(32)	Yes	..
address1_street	varchar(64)	Yes	..
address1_ postalcode	varchar(32)	Yes	..
address2_city	varchar(32)	Yes	Second address
address2_country	varchar(32)	Yes	..
address2_number	varchar(32)	Yes	..
address2_number_ addition	varchar(32)	Yes	..
address2_ postalcode	varchar(32)	Yes	..
address2_street	varchar(32)	Yes	..
contact_street	varchar(32)	Yes	Contact (like parents etc.)

Continues on next page.

A. User (cont.)

Values and requirements. Check current use of field before use.

Attribute	Type	Optional	Description
contact_city	varchar(32)	Yes	..
contact_phone1	varchar(32)	Yes	..
contact_country	varchar(32)	Yes	..
contact_email	varchar(32)	Yes	..
contact_number	varchar(32)	Yes	..
contact_number_addition	varchar(32)	Yes	..
contact_phone2	varchar(32)	Yes	..
gallery_id	varchar(32)	Yes	(Legacy)
insert_date	timestamp	Yes	(auto)
update_date	timestamp	Yes	(auto)
notes	mediumtext	Yes	..
notes_extra	mediumtext	Yes	..
type	varchar(32)	Yes	Empty string allowed
disabled	tinyint(1)	Yes	0 / 1 (1=Disabled) Default 0
vo_notification	tinyint(1)	Yes	0 / 1 (0=Disabled) Default 1
date_updated	timestamp	Yes	Last time updated
nationality	varchar(32)	Yes	
custom_fields	text	Yes	Json metadata, required is some student app forms

A. Poll

Values and requirements. Check current use of field before use.

Attribute	Type	Optional	Description
question	varchar(250)	No	Question of the poll
start	datetime	No	Startdate Poll
end	datetime	No	Enddate Poll
auth_id	int(11)	Yes	Auth id of user created image
show_on_wall	tinyint(1)	Yes	Allow showing on wall (0=Disabled)
external_id	varchar(254)	Yes	External Identifier
hide_answer	tinyint(1)	Yes	Allow showing off answers (0=Disabled)

A. Eventcalendar

Values and requirements. Check current use of field before use.

Attribute	Type	Optional	Description
event_id	int(11)	No	API id of an event object
participation	int(11)	No	Number of current participation
max_participants	int(11)	No	Max. number of participants
calendar_id	int(11)	Yes	API id of a calendar object
subscribe_start	datetime	No	Start date and time of subscription period
subscribe_end	datetime	No	End date and time of subscription period

A. Event

Values and requirements. Check current use of field before use.

Attribute	Type	Optional	Description
title	varchar(128)	No	Title of Event. Empty string allowed
day	date	No	Start date and time of event
text	mediumtext	No	Text explaining event. Empty string allowed
image	varchar(64)	No	URL, Image will be buffered in local image server. Empty string allowed.
external_id	varchar(254)	Yes	External Identifier
date_updated	timestamp	Yes	Last time updated
type	enum('time', 'date')	No	Set event as a time (like lunch/happy hour) or date (like anniversary) focused event
guest_message	mediumtext	No	Guest message for guest user type if guesguest_mode is enabled. Empty string allowed.
has_info	tinyint(1)	No	Boolean type option. Set to 1 (one) to set the presence of info
info	mediumtext	No	Extra information if has_info is enabled. Empty string allowed.
speakers	tinyint(1)	No	Boolean type option. Set the presence of info
guest_mode	tinyint(1)	No	Boolean type option. Set to 1 (one) to use guest_mode
info_url	mediumtext	No	Link to extra info page. Empty string allowed.
group	int(11)	No	API id of an group object. Used to assign event to a group. 0 (zero) is allowed
survey_url	mediumtext	No	Link to extra survey. Empty string allowed.

A. Event (cont.)

Values and requirements. Check current use of field before use.

Attribute	Type	Optional	Description
event_image_header_image	varchar(64)	Yes	URL, Image will be buffered in local image server. Empty string allowed.
event_image_background_image	varchar(64)	Yes	URL, Image will be buffered in local image server. Empty string allowed.
speaker_name	varchar(64)	Yes	Name of speaker
auth_id	int(11)	Yes	Auth id of event organizer. default: NULL

A. Gallery

Values and requirements. Check current use of field before use. The addition of a Auth (Authentication) object or the retrieval of Auth information via the user calls

Attribute	Type	Optional	Description
title	varchar(64)	No	Title of Event. Empty string allowed
rights	int(11)	No	Who will be able to view the data. 0 (zero) no one, 1 (one) Guests, 2 (two) Users. Works as binary
text	mediumtext	No	Text explaining event. Empty string allowed
image	varchar(64)	No	URL, Image will be buffered in local image server. Empty string allowed
auth_id	int(11)	Yes	Auth id of user image author
external_id	varchar(254)	Yes	External Identifier
date_updated	timestamp	Yes	Last time updated

A. Galleryimages

Values and requirements. Check current use of field before use. The addition of a Auth (Authentication) object or the retrieval of Auth information via the user calls

Attribute	Type	Optional	Description
title	varchar(128)	No	Title of Event. Empty string allowed
gallery_id	int(11)	No	Id of API element gallery that contains this image
text	mediumtext	No	Text explaining event. Empty string allowed
url	varchar(128)	No	URL, Image will be buffered in local image server. Empty string allowed
auth_id	int(11)	No	Auth id of user created image
external_id	varchar(254)	Yes	External Identifier
date_updated	timestamp	Yes	Last time updated

A. Poll/Votes

Values and requirements. Check current use of field before use.

Attribute	Type	Optional	Description
poll_id	int(11)	No	Id of API element: poll
user_id	int(11)	No	Id of API element: user
created_at	timestamp	Yes	Time created
updated_at	timestamp	Yes	Last time updated

A. Poll/Option

Values and requirements. Check current use of field before use.

Attribute	Type	Optional	Description
poll_id	int(11)	No	Id of API element: poll
value	varchar(200)	No	Date and time of last vote
external_id	varchar(254)	Yes	External Identifier
created_at	timestamp	Yes	Time created
updated_at	timestamp	Yes	Last time updated

A. Event/Images

Values and requirements. Check current use of field before use.

Attribute	Type	Optional	Description
title	varchar(128)	No	Title of image. Empty string allowed
event_id	int(11)	No	Id of API element: event
image	varchar(128)	No	URL, Image will be buffered in local image server. Empty string allowed
created_at	datetime	Yes	Time of upload
updated_at	datetime	Yes	Last time updated

A. Event/Speaker

Values and requirements. Check current use of field before use.

Attribute	Type	Optional	Description
name	varchar(128)	No	Full name of the speaker
event_id	int(11)	No	Id of API element: event
image	varchar(128)	No	URL, Image will be buffered in local image server. Empty string allowed
description	mediumtext	No	Description and background of speaker
created_at	datetime	Yes	Time of upload
updated_at	datetime	Yes	Last time updated

A. Eventcalendar/Participant

Values and requirements. Check current use of field before use.

Attribute	Type	Optional	Description
kicked	tinyint(1)	Yes	Registered the removal of the participant. 1 (One) means user is kicked. Default is 0 (Zero)
event_calendar_id	int(11)	Yes	Id of API element: eventcalendar
calendar_id	int(11)	No	Id of API element: calendar
auth_id	int(11)	No	Auth id of user (participant)
created_at	datetime	Yes	Time of upload
updated_at	datetime	Yes	Last time updated

A. Eventcalendar/Lineup

Values and requirements. Check current use of field before use.

Attribute	Type	Optional	Description
title	varchar(128)	No	Title of Even line-upt. Empty string allowed
event_calendar_id	int(11)	Yes	Id of API element: eventcalender
image	varchar(64)	No	URL, Image will be buffered in local image server. Empty string allowed
artist	varchar(64)	No	The name of the artist, band or public speaker
description	mediumtext	No	Description of this event lineup
start	datetime	No	Start of this lineup element
end	datetime	No	End of this lineup element
date_updated	timestamp	Yes	Last update on this object
external_id	varchar(254)	Yes	External Identifier

A. Eventcalendar/Speaker

Values and requirements. Check current use of field before use.

Attribute	Type	Optional	Description
event_calendar_id	int(11)	No	Id of API element: eventcalendar
speaker_id	int(11)	No	Id of API element: event