Edupage API

Communication overview

Communication is done through simple HTTP requests. These requests can be either GET or POST requests and all parameters can be passed in both ways: in URL query string or in POST content or any combination of both.
Note: There is a limit on the size of the URL in the web server, so for parameters that can be very long it is recommended to send them as POST.
Note 2: If you send data in POST, the mime type should be "application/x-www-form-urlencoded" or similar.

Communication URL has this form:

https://**edupage_name**.edupage.org/eduapi

Where "edupage_name" is the name of the school's edupage (each school has a different edupage_name). Rest of the URL is fixed.

Required parameters

apikey

This is a kind of password for accessing Edupage API for a school's Edupage page. One school (and one Edupage web page) can have multiple API keys. It is possible to configure access rights for every API key individually, so the school can give one API key to the company providing attendance system (with postbeeps right) and other to the company providing digital signages (with getdailyplan right).

API keys are created/managed in Edupage by administrator in Agenda Online - Tools - Administration - Edupage API.

Preferred way to send API key is to use the **Authorization** header. You can also send it as a POST parameter.
Note (2025/09): Edupage API will accept API key also as URL query parameter, but this is strongly discouraged and will be removed in the future as this way the API key may get into log files.

cmd

This parameter specifies the API command. Commands are listed later in a separate chapter.

Responses

Success

Exact format of response depends on individual command, but usually commands return XML or JSON files on success. XML response will contain a "status" attribute in its root node, JSON response will have "__eduapi_status" key in root object. When status="OK", it indicates success

in processing the command. Another typical value is status=”WARN”, which means that the command succeeded, but there were some minor problems. These problems will be listed within the rest of XML/JSON.

Error handling

On error, API requests usually return a string response starting with ErrorXXXXXXXX, where XXXXXXXX is the error number. This will be followed by a text explanation.

WAIT response

There is a certain limit on how often you can do API requests with one API key. If you access the API too frequently, you may receive text response starting with WAIT. This means that you should wait a certain period before calling API again. Recommended wait time is 10 seconds.

Blocked because of abuse

It is important that you monitor reponses from API and check for errors. Keep log of responses on your side. If your use of Edupage API will cause many errors on our servers (or many WAIT responses), it may lead to automatic blocking of API key. These blocks usually last for 1 week.

If you use some background process that synchronizes data from Edupage to your system in a fixed time interval (API polling), avoid using intervals shorter than 5 minutes.

Commands

getbasedata

With this command you can get basic data from Edupage. This includes a list of teachers, classes, subjects, classrooms and students. Parameters:

- **format** - xml (default) or json (recommended)

The command operates in 2 modes: date or year. This is controlled by sending one of these parameters:

- **date** - date for which you are getting data. This will return only records:
    - valid for provided date - e.g. the date is not outside of datefrom/dateto interval of the record.
    - with values for that date - e.g. if student changes class during school year, in classid you will get a class in which he is for the given date
- **year** - school year for which you are getting data (for countries with summer holidays send lower calendar year, eg. for 2017/2018 send just 2017). This will return records:
    - for the whole school year (including those that are not valid for a current day = outside of datefrom/dateto interval).
    - fields that can change value during the school year will have the newest value.

If you do not provide date or year parameter, the default is date=today.

“id” fields in returned data are internal identifiers in Edupage database. These are non-empty strings of printable ASCII characters. They are unique within individual table.

Note: access to some tables (e.g. students) or columns (e.g. datefrom/dateto) must be enabled in settings of API key (this must be done manually by school in settings of Edupage API)

getphotos

With this command you can get photos of students or teachers from Edupage. Parameters:

- **studentids** - ids of students for which you want to get photos
- **teacherids** - ids of teachers

Note: There is a limit on the total amount of data transferred by this request. If this limit is exceeded, you will get photos only for a part of students/teachers and status will be ”OK_part”. You will have to repeat the “getphotos” command again with the rest of the students/teachers.

getisicorders

This command is for issuers of ISIC/ITIC cards. Schools can mark students/teachers that need a new card with value “New” in the “ISIC - Order” field and the issuer can then get data needed to issue cards using this API command.

With this command you can get data about students/teachers that have some value in the “ISIC - Order” field. The data is returned in the same format as with the “getbasedata” command, but includes only data for students/teachers with some value in ISIC/ITIC order and automatically includes all columns necessary for issuing ISIC/ITIC cards.

Note: If you do not see “ISIC/ITIC - Order” field in student or teacher dialog in Agenda Online, it may be possible that this field is not enabled in your country. In such case, please contact us.

postbeeps

With this command you can post data from the attendance system to Edupage. Data is sent through parameter **data**, which should contain XML file like this:

```
<beeps>
  <beep time="2015-03-16 09:40:03"  device=”01:23:45:67:89:ab” ecard="12345678"
type=”entrance” studentid="123" />
  <beep ... />
  <beep ... />
  <beep ... />
</beeps>
```

Each beep row has these parameters:

- **time** (required) - time of beep in local time zone. Format is “YYYY-MM-DD HH:MM:SS”. Valid range is from 7 days in the past till 5 minutes in the future (against Edupage server local time).
- **device** (required) - internal identification of device used to scan id token. Device numbers should be unique and there should be a finite number of them (e.g. you can not have one device per teacher/student). Recommended value formats are:
    - MAC address of the device.
    - In case there are multiple devices connected to the main unit, you can append dot and port number.letter, like “01:23:45:67:89:ab**.1**”
    - In case you want to use some internal number from your system to identify the device, prefix it with the name of your company and dot, like “OUR_COMPANY.1234”. This is needed to ensure that the device is unique in case school uses devices from multiple companies.
- **ecard** (optional) - id token number (e.g. barcode or RFID number)
- **studentid** or **teacherid** (required if ecard was not provided, otherwise optional) - this should be a valid id number from the list of students or teachers returned from getbasedata command.
- **type** (optional) - if your device can distinguish between entrance and exit, or other type, you can identify it here.
    - **entrance** - Person has entered school building
    - **exit** - Person has left school building
    - **vacation**, **doctor**, … - If your device supports identifying reasons for leaving the building or some other info, you can send custom tags for that in type attribute.These custom tags must exactly match with name or short name of some entry from Edupage - Substitution - Settings - Reasons of absence.
    - You can also combine multiple values in this field with a comma, e.g. type=”exit,doctor”.

It is recommended to send data in 1 minute intervals. Avoid sending 1 beep per API call as you will probably hit API calls limit (WAIT response) during “rush hours”. Instead send all pending beeps in one request.

If you will accidentally send the same beep in multiple API requests, there is no problem with that. It will just overwrite previously sent beep (with the same type and for the same user).

You can check successfully posted beeps in Edupage - Start - Security - Arrivals to school - Show list of arrivals.

Note: It is recommended to send beeps as POST parameter instead of GET parameters, because GET parameters are sent in URL and there is a limit on URL size in the web server.

getdailyplan

Gives you a list of lessons and other events that happen on a given day. These may come from timetable, substitutions and other sources within Edupage. Parameters:

- **date** - date for which you are getting data (optional)
- **datefrom**, **dateto** - get data for more days with one request (optional). You can get data for up to 7 days in a single request.
- **purpose** - (required) Value of this parameter depends on what kind of data you want to get (for which purpose you want to use it). Please contact us about proper value for this parameter for your case.

getroomoccupancy

This function gives you all the data needed to implement an automated room heating system. It combines results from **getdailyplan** and **getbasedata** commands and limits the data to only those parts that are needed by room heating systems. Parameters:

- **date** or **datefrom**, **dateto** - see description in **getdailyplan** command.

listtimetables

Gives you list of timetable files from Timetables Online. Returns JSON.

gettimetable

Download timetable data. Parameters:

- **timetableid** (required)
- **format** (required):
    - **json** - Recommended format for new integrators. See TimetableJson.
    - **asctt2012.xml** - Asc Timetables 2012 XML format
    - **asctt2008.xml** - Asc Timetables 2008 XML format

- **idmode**
    - **timetable** (default) - return ids as if the file was exported from aSc Timetables PC application (no linking to Edupage/SIS data)
    - **edupage** - for items linked to Edupage return edupage id, for others return raw ids.
    - **sis** - for items linked to Edupage that have also sisid return sisid, for others return raw ids.
    - **ascttchrid** - available only with format=json.

Notes:

- For XML formats, function returns “raw” exported XML data, so there will be no “status” field in return resposne.
- If you use idmode other than ‘timetable’, you may sometimes get in one table multiple rows with the same ‘id’. E.g. for subjects it is possible in Edupage to “link” two timetable

subjects to one edupage subject. Duplicate ids may happen also if there are multipe rows in edupage database that have the same ‘sisid’.

logintopartner

With this command you can implement automatic login from Edupage to an external system.

Setup on school’s side:

School has to add this special link somewhere on their Edupage web site:

/eduapi?cmd=logintopartner&url=EXTERNAL_SYSTEM_URL

They can use one of these modules to add a link:
Menu (top, left, right)
Text module
News item
(or any other Edupage web site module that supports links)

How it works:

If logged user clicks on a link, he/she gets redirected to your page at EXTERNAL_SYSTEM_URL with added these 2 url parameters:

- **edupage** - school from which this redirect comes.
- **logintoken** - newly generated token that encodes currently logged user to edupage. You can decode this token by calling API command “logintopartner”. Note: For security reasons this token is valid only for a very short period of time (e.g. 15 seconds), so your web server must immediatelly call edupage API to resolve it.

“logintopartner” API command

Parameters of Edupage API command:

- **logintoken** - Login token you have received from the redirect link from Edupage.

Returns JSON with these fields:

- **url** - This is the url parameter from redirect link (explained above). It is important to check if it matches your system’s url and reject logintoken with different url.
- **admin** - If user was logged as administrator to Edupage
- **teacherid** - If user was logged as teacher to Edupage
- **studentid** - If the user was logged as student to Edupage, or if user was logged as parent of the student.
- **parentid** - If the user was logged as parent to Edupage.

noticeboard_writepdf

Updates pdf file shown in PDF noticeboard. Parameters:

- **nb** - your noticeboard number.
- **data** - data of pdf file.

Example using CURL command line tool:

```
curl -k -v --data-urlencode "apikey=YOUR_API_KEY" --data-urlencode
"cmd=noticeboard_writepdf" --data-urlencode "nb=YOUR_NB_NUMBER"
--data-urlencode "data@YOUR_PDF_FILE.pdf"
https://YOUR_EDUPAGE.edupage.org/eduapi
```

noticeboard_writeimage

Updates image file shown in Image noticeboard. Parameters:

- **nb** - your noticeboard number.
- **data** - data of image file.

~~Depreacated: substitution_import~~

2025-02-01 This function is no longer supported

Automatize import function for substitution (when school is using other timetabling program and importing data to Edupage). Parameters:

- **data** - data from import file
- **multi** - (optional) 1 - do multi day import
- **publish** - (optional) 1 - immediately publish on successful import

Edupage data structures

Here you can find documentation about some Edupage internales that are important for using Edupage API.

School years

In Edupage each date is assigned to a fixed school year. This is based on the so-called “turn over day” setting. For schools in the northern hemisphere this is usually 08-01 (August 1st) and in the southern hemisphere it is 01-01 (January 1st), but it may vary from country to country and even within countries, especially for countries close to the equator. This setting is not user configurable. Its default value is based on country, but it can be changed to a different day upon school/partner request.
Internally school year is represented by int value **year** - it is equal to year of the first day within that school year. This means that e.g. year 2023/2024 (for northern hemisphere country) is represented as year=2023.

Year is very important for some parts of Edupage database - especially for table of **classes**. In most schools classes change names with a new school year, e.g. 5A becomes 6A or 501 becomes 601, etc. This means that e.g. when you call **getbasedata** command for 2 consecutive years (parameter **year=**...), you will get a different name for the same class id. You will also get a different list of classes, as some classes may have finished education and others may have just started.

Year is also important for other tables.

- For **students** you will always get only a list of students that are present in the year. Students will have the same id in between years, but some fields may have different values each year (e.g. **grade**).
- For **teachers** we have fields **datefrom** and **dateto** that indicate start/end of employment of that teacher. **getbasedata** will not return teachers that are outside of the school year.
- For some tables, like **subjects** and **classrooms**, we have a field **yearto** (in UI it is called “Used till year”).

Timetable database vs Edupage database

Each timetable file in Edupage is a separate document with its own database of subjects, classes, teachers, etc. and this is separate from Edupage database. Timetable database is accessed with **gettimetable** command and Edupage database with **getbasedata** command. All other API commands work with Edupage database. This also means that timetable database has its own ids distinct from ids from Edupage database.
Note: If timetable is properly linked to edupage, you can control which ids you get from **gettimetable** command with **idmode** parameter.

Timetable ids

Native timetable ids start with * character, e.g. *1, *2, *3… They are valid only within one result of **gettimetable** call. If the school has multiple timetable files, same object can have in each file a different timetable id. And these ids may also get reassigned when user changes individual timetable. This means that you can not use these ids for long term linking of timetable data to your external data.
For linking with external data some tables in timetable database have an additional column **ascttchrid**. These look like 16 digit hex numbers e.g. 0D136D21DB6CCE9D, but in fact they are just a longer randomly generated strings. They may also get reassigned when user changes the timetable, but in contrast with native timetable ids they will never “move” to a different object. Edupage uses these internally for linking between timetable and edupage database.

Assignment of timetables to dates

Schools can have any number of timetable files stored in Edupage. Some of these timetables may be published as official timetables. They will have **state=official** and filled corresponding **year** and **datefrom**/**dateto** fields in **listtimetables** result. There may be multiple official (=published) timetables within one school year, but there can be only one timetable valid for each day. If there are multiple timetables published with overlapping **datefrom/dateto** intervals, then the one with higher **datefrom** is used. You can see which timetable is valid for a given day by calling **getdailyplan** command and checking **timetableid** field for each date.

Explanation: Schools often change timetable during the school year to accommodate some unexpected change (e.g. some teacher has left the school, or some classroom becomes unavailable) or they may use completely different timetable for parts of the year (e.g. for 1st and 2nd term). Edupage needs to keep track of these changes and know which timetable was valid for each date. This is important especially for substitution and class register functions. So if there is a change during the school year, school should not edit existent timetable, as this may lead to data corruption (e.g. past substtiton data has to match with timetable data). They have to duplicate existing timetable file (using “Save as…” function) and then publish this new timetable file with a new **datefrom** value.

TimetableJson

Below you can find type definition for response from **gettimetable** command in **format=json**. This shows just the basic structure, to better understand the structure please use the API to get json for an actual timetable for some of your schools.
Note: The format includes just a basic set of fields needed by most integrators. If you need some additional timetable data not present in the basic format, please contact us.

```
type TimetableJson = {
  timetableid: string;
  periods: {
```

```
  rows: {
    id: string;
    name: string;
    short: string;
    starttime: string;
    endtime: string;
    daydata: {
      [key: string]: {
        starttime: string;
        endtime: string;
      };
    };
  }[];
};
breaks: {
  rows: {
    id: string;
    name: string;
    short: string;
    starttime: string;
    endtime: string;
    daydata: {
      [key: string]: {
        starttime: string;
        endtime: string;
      };
    };
  }[];
};
days: {
  rows: {
    id: string;
    name: string;
    short: string;
  }[];
};
weeks: {
```

```
  rows: {
    id: string;
    name: string;
    short: string;
  }[];
};
daysdefs: {
  rows: {
    id: string;
    name: string;
    short: string;
    typ: string;
    vals: string[];
  }[];
};
weeksdefs: {
  rows: {
    id: string;
    name: string;
    short: string;
    typ: string;
    vals: string[];
  }[];
};
termsdefs: {
  rows: {
    id: string;
    name: string;
    short: string;
    typ: string;
    vals: string[];
  }[];
};
subjects: {
  rows: {
    id: string;
    name: string;
```

```
    short: string;
  }[];
};
teachers: {
  rows: {
    id: string;
    firstname: string;
    lastname: string;
    short: string;
    gender: string;
    color: string;
  }[];
};
buildings: {
  rows: {
    id: string;
    name: string;
    short: string;
  }[];
};
classrooms: {
  rows: {
    id: string;
    name: string;
    short: string;
    buildingid: string;
  }[];
};
classes: {
  rows: {
    id: string;
    name: string;
    short: string;
    classroomid: string;
  }[];
};
groups: {
```

```
    rows: {
      id: string;
      classid: string;
      name: string;
      entireclass: boolean;
      divisionid: string;
    }[];
  };
  lessons: {
    rows: {
      id: string;
      subjectid: string;
      classids: string[];
      groupids: string[];
      seminargroup: string | null;
      teacherids: string[];
      count: number;
      durationperiods: number;
      daysdefid: string;
      weeksdefid: string;
      termsdefid: string;
      terms: string;
    }[];
  };
  cards: {
    rows: {
      id: string;
      lessonid: string;
      periodid: string;
      dayid: string;
      weekid: string;
      classroomids: string[];
    }[];
  };
};
```