
OpenStudio Docs Documentation

Edwin van de Ven

Jan 04, 2019

1	System requirements	1
2	Web2py versions	3
3	Installation	5
4	Update OpenStudio	11
5	Quickstart	13
6	Customers	19
7	Manage subscription credits	21
8	Manage class access	23
9	Teacher payments	25
10	MailChimp	29
11	Exact Online	31
12	API Reference	33
13	Right of consent	45
14	Right to access	47
15	Right to be forgotten	49
16	License & Disclaimer	51

System requirements

Please keep in mind that OpenStudio is a web application which needs more resources than a standard website.

1.1 Hardware

Minimum

- 2 (v)CPU cores*
- 2 GB of RAM
- 1 GB of free storage

Recommended

- 2 (v)CPU cores*
- 4 GB of RAM
- 10 GB of free storage
- SSD storage

* With performance comparable to at least a second generation Intel Core i5 @ 2.0 Ghz

1.2 Hosting platform

While in theory everything used to build OpenStudio should run on Windows and MacOS as well, it's only tested on Linux. For a production system, this is probably the best choice.

General

- Python 2.7 (Not Python 3)
- MySQL 5.5 or later database

- Redis server (Tested using version shipped with Ubuntu 18.04 - 4.0.9-1)
- Web2py (<http://www.web2py.com>)

Optional but recommended for a production setup

- Nginx
- uWSGI
- SSL Certificate

You might want to have a look at [Let's Encrypt](#) and support the project if you can.

Python Modules v2018.82 and later

- openpyxl
- html2text
- pytz
- redis (2.10.6)
- mollie-api-python (2.x)
- weasyprint (0.42.3)
- Pillow
- pybarcode
- qrcode
- mailchimp3

Python Modules v2018.81 and earlier

- openpyxl
- html2text
- pytz
- redis (2.10.6)
- mollie-api-python 1.4.2 (not compatible with 2.x)
- weasyprint (0.42.3)
- Pillow
- pybarcode
- qrcode
- mailchimp3

CHAPTER 2

Web2py versions

- OpenStudio 2018.82.0 - web2py 2.17.2
- OpenStudio 2018.81.0 - web2py 2.17.1
- OpenStudio 2017.x up to 2018.8.x - web2py 2.15.4

This is a short guide to get OpenStudio running on Ubuntu 18.04.1 LTS Desktop as quickly as possible. This setup is not recommended for production usage. For production usage you'd want to use something like nginx together with uwsgi and make sure you have a valid SSL certificate. ** Using this system without HTTPS will transmit unencrypted (easily interceptable and readable) information over the networks between you and the computer hosting the application. **

3.1 General

A note about time

OpenStudio assumes system time is set to UTC and the timezone will be configured in Settings - System - General. All (date)times in scheduled tasks are also set in UTC.

A note about operating systems

OpenStudio is developed and tested on Linux. In theory Windows and MacOS should be able to serve as a hosting platform as well. On paper all components should be compatible. Though note that this is untested and unsupported, so your mileage may vary.

3.2 Operating System

This installation manual is based specifically on deployment to Ubuntu 18.04.1 LTS Desktop which can be located here: <https://www.ubuntu.com/download/desktop> . The Desktop distribution is used to simplify initial administration of Web2Py server which REQUIRES a secure connection either locally, remotely over SSH tunnel or HTTPS. SSH tunnels and HTTPS are not covered in this manual. Install said operating system to a computer or virtual machine before proceeding to follow these instructions.

While using said OS and especially while using a different OS, you may encounter different results or barriers to installation that require additional effort to overcome before proceeding to the next step. Experience with the use of Linux operating system is assumed and recommended.

3.3 Database

In this manual we are deploying MySQL 5.7. You can also use another database server, such as MariaDB. Depending on which distribution or platform you are using there are different ways to install MySQL server. On Ubuntu Linux use the following command:

```
sudo apt-get install mysql-server-5.7
```

For Windows or Mac OS, please see this page to download the free version <http://dev.mysql.com/downloads/mysql/> . Install it according to the manual of your operating system. After installing the MySQL server, make sure the MySQL service is started and connect to the database using the mysql command-line tool.

```
sudo mysql -u root -p
```

Then create a database for OpenStudio, in this example the database name `openstudio_db` will be used.

```
mysql> create database openstudio_db;
mysql> grant all privileges on openstudio_db.* to 'your username goes here'@'localhost
↳' identified by 'your password goes here';
mysql> flush privileges;
mysql> exit
```

The output should look something like this assuming your linux username is 'user':

```
mysql> create database openstudio_db;
      Query OK, 1 row affected (0.00 sec)

mysql> grant all privileges on openstudio_db.* to 'user'@'localhost' identified by
↳'password';
      Query OK, 0 rows affected, 1 warning (0.01 sec)

mysql> flush privileges;
      Query OK, 0 rows affected (0.00 sec)

mysql> exit
Bye
```

That's it, now test the database by connecting using the command-line tool.

```
mysql -u<username> -p<password> openstudio_db
```

If all goes well, you'll see the `mysql>` prompt again. Then exit the `mysql` command-line tool and resume with the next step. If you got an error message, troubleshoot and solve it before moving on.

3.4 Redis Server

We need to install the database caching server. This can be done in Ubuntu Linux by typing the following command:

```
sudo apt-get install redis-server
```

For Windows or Mac OS, you can download Redis Server from <https://redis.io/download> Follow the application vendors instructions for installing and running Redis on Windows or Mac.

3.5 Python 2.7

For Windows you can use Python 2.7.12 (or later), pip ships with this version (get it from www.python.org).

For MacOS it's likely there's already a python installation, but it might be old if you're running an older version of MacOS. So you might want to use macports (or whatever way you prefer) to install the python27 port. (www.macports.org) Updating the system installed python version is not recommended and you are doing so at your own risk.

Install Python 2.7 on Ubuntu Linux by issuing the following commands:

```
sudo apt-get install python
sudo apt-get install python-pip
```

Install the following Python Modules:

openpyxl, html2text, pytz, redis (2.10.6), mollie-api-python (2.x), weasyprint (0.42.3), Pillow, pybarcode, qrcode, mailchimp3

In order to install said modules in Ubuntu Linux, issue the following commands:

```
sudo -H pip install openpyxl
sudo -H pip install html2text
sudo -H pip install pytz
sudo -H pip install redis==2.10.6
sudo -H pip install mollie-api-python==2.0.6
sudo -H pip install weasyprint==0.42.3
sudo -H pip install Pillow
sudo -H pip install pybarcode
sudo -H pip install qrcode
sudo -H pip install mailchimp3
```

3.6 Web2py

Now download web2py from www.web2py.com and extract it to a directory that's suitable for you. In this example I'll use a separate folder in the Home Directory called www-dev

```
sudo mkdir /home/www-dev
sudo chmod 777 /home/www-dev
cd /home/www-dev
wget https://mdipierro.pythonanywhere.com/examples/static/web2py_src.zip
unzip web2py_src.zip
```

Please note that setting directory permissions to 777 should never be done in production

Now we need to download and extract the latest OpenStudio release archive (zip or tar.gz) to the applications folder in your web2py installation. The latest release can be found here: <https://github.com/openstudioproject/openstudio/releases>

In this manual version 2018.84.2 will be assumed. After extraction we need to rename the extracted folder to 'openstudio' If you neglect to rename the extracted folders, the special characters will prevent the application from being detected by web2py. In order to do so in Ubuntu Linux, issue the following commands:

```
cd /home/www-dev/web2py/applications
wget https://github.com/openstudioproject/openstudio/archive/v2018.84.2.zip
```

(continues on next page)

(continued from previous page)

```
unzip v2018.84.2.zip
mv openstudio-2018.84.2 openstudio
```

Now you can start web2py by opening a terminal and browsing to the directory you extracted web2py in and then using `python run web2py`.

To start web2py on Ubuntu Linux, issue the following commands:

```
cd /home/www-dev/web2py
python web2py.py -a <choose a password>
```

Windows

```
c:\python27\python.exe web2py.py -a <choose a password>
```

Open a web browser ON THE HOST COMPUTER (this is why we've installed desktop gui in this guide) and browse to <http://localhost:8000> Now click on the hamburger button (The three horizontal lines for menu) on the top right of the page and click 'My Sites'. You should have openstudio in the list of Installed applications on the left of the page. If you don't, check that the directory name of the openstudio folder under `/web2py/applications/` doesn't have any special characters in it and restart web2py.

Click the manage button next to OpenStudio and select Edit from the drop down list that appears. Near the bottom of the list in the Private Files section of the edit page, click Edit to the left of `appconfig.ini` - Here is a line that needs to be edited.

```
uri = mysql://user:password@localhost/openstudio_db
```

In a previous step we created a MySQL database to hold all the information. The uri option in this file tells OpenStudio how to connect to the MySQL database. You need to replace 'user:password' with the username and password you authorized during the create database step.

If you installed the MySQL server on the same computer, you can use 'localhost' as the server name. After editing the file, scroll to the top of the page and save the file (you can click the floppy disk icon or also use Ctrl+S). If already started, Web2py will need to be restarted after editing `appconfig.ini`, the settings are only read when the framework is started.

At this stage the application is available for login from the host computer at <http://localhost:8000/openstudio> - HOWEVER, it is liable to malfunction unpredictably until you configure a `routes.py` file. By default, when web2py runs, it binds to 127.0.0.1. If you want to be able to access the application remotely from the host computer, you need to start web2py with additional argument `-i {ip address}`. You can define the binding port with `-p {port number}`. You cannot administrate your sites in web2py admin unless you are running it localhost and access from same computer or remote computer via SSH tunnel OR bind to a routeable IP and access using HTTPS.

Starting from version 2.07 Javascript (AJAJ) is used more to make the interface more user friendly. However to make it work, you should use a `routes.py` file in your web2py root folder to be able to run openstudio from an url like "<http://demo.openstudioproject.com>". The url shouldn't have the app name in it, a url like "<http://localhost:8000/OpenStudio>" will cause problems. The `routes.py` file can look like this for example:

```
routes = dict(      # base router
    BASE = dict(
        default_application = 'openstudio',
        domains = {
            'demo.openstudioproject.com' : 'openstudio',
        },
        applications = ['openstudio', 'admin'],
        controllers = 'DEFAULT'
```

(continues on next page)

(continued from previous page)

```
) ,  
)
```

After adding the routes.py file in the web2py root folder, restart web2py. Make sure your DNS records or hosts file point to the correct name.

3.7 Scheduler

Starting from version 2018.82 the Web2py Scheduler is required to use all features in OpenStudio. Please refer to the Web2py book for instructions on how to set up the scheduler: [Web2py book](#).

3.8 Logging in

Go to the address where you're hosting OpenStudio. If everything went well, there will be a login screen.

Default username and password The default username and password are admin and admin for versions lower than 2.05. For version 2.05 and newer, the default username and password are admin@openstudioproject.com and admin. For version 3.0 and newer, the default username and password are admin@openstudioproject.com and OSAdmin1#.

Now you're ready to start.

3.9 Troubleshooting

In case you see an error like the one below, please check that the python interpreter you're using to run OpenStudio can find the python modules mentioned in the system requirements.

```
"Cannot import module 'applications.openstudio.modules.pytz'"
```

Update OpenStudio

Below are the steps to upgrade OpenStudio 2017 (and later), for example from version 2017.1.0 to 2017.2.0. The steps are the same for all upgrades.

1. Create a backup of your web2py folder and your openstudio database.
2. Check your backups, do they contain data and are you able to restore them to another location. (If you're not sure you can restore it, it doesn't count as a backup)
3. Check the web2py versions section in this manual to see if the web2py version is the same for the new version. In case a newer web2py version is required, download it and set it up. For more info on setting up web2py, please see the web2py documentation.
4. In case you don't have to upgrade the web2py installation, just remove the OpenStudio app using the web2py admin interface.
5. Extract the latest OpenStudio version in the web2py applications folder.
6. Next edit the application in the admin interface and edit appconfig.ini in the private folder to connect OpenStudio to the existing database.
7. Copy the databases folder, uploads folder and custom folders in views/templates from your web2py/applications/<OpenStudio_backup> to web2py/applications/<OpenStudio updated>.
8. Visit your openstudio url and add /upgrade at the end. eg <https://demo.openstudioproject.com/upgrade>
9. Open the OpenStudio url and you should be able to log in to the new version.

In case you need help, paid support is offered.

5.1 Installation

Please go to [readthedocs](#) for the latest installation manual

5.2 Getting Started

The first things you'll want to do to get started with OpenStudio is set up your school. You can access the school properties using the school menu at the top of the screen. Below is a list of the different available properties and their purpose.

5.2.1 1. Teachers

Before classes can be added to the schedule, at least one teacher should be specified.

In the teachers list there is a class types button for every teacher. Using this button you can set the types of classes this teacher is willing and able to teach. Later when adding or editing a class and selecting a teacher who is not marked as a teacher for that class type, a notification will be shown when saving the class.

5.2.2 2. Employees

Here you can configure and add Employees of your school.

5.2.3 3. Memberships, Subscriptions & Class Cards

Here you configure the different kinds of memberships, subscriptions and classcards offered by your school.

5.2.4 4. Class Types

Here you can specify what types of classes are taught at your school. You should add at least one class type to be able to add classes later.

5.2.5 5. Locations

Using locations you can specify the location of classes. You should add at least one location to be able to add classes later.

5.2.6 6. Shifts

Office and Employee Shifts are defined here.

5.2.7 7. Holidays

Need a break? For each location you can specify when you're having a holiday and OpenStudio will automatically cancel all classes during that period.

5.2.8 8. Languages

Languages are defined here. You can link languages to customers as a reminder for yourself in which language you should communicate with them.

5.2.9 9. Practice Levels

This creates a list of levels that can be applied to customers and classes. For example you might want to keep track of whether customers should take introduction courses or not or whether certain classes are appropriate for them.

5.2.10 10. Discovery

Discovery is used to specify how customers found your school. Here you could for example add things like "Google", "Referral by a friend" or "Advertisement spring 2013". What you enter here will appear in a drop down list in the edit customer pages. After filling out this field for at least one customer you can go to statistics -> discovery and see a chart of the distribution of the different ways customers have found your school. This allows you to keep track of the efficacy of different kinds of advertisement and marketing.

5.2.11 11. Keys

There is nothing to fill out here. When the field "Key Number" is filled out when adding or editing a customer, the customers appear in this list sorted by key number. This page can be used to quickly get an overview or search for keys when your school distributed keys, swipe cards or anything like that to customers so they can enter your school

5.3 Schedule

Use the schedule to keep track of your classes, attendance for those classes, holidays and more. Furthermore you can accept and decline substitute requests for classes and manage the schedule for Studio staff.

5.3.1 1. Classes

1.1. Adding Classes

To add a new class to the schedule, click the add “Add a new class” on the right. To quickly add similar classes, click the edit button (the one with the pencil icon) on the Schedule page and then use the duplicate link to duplicate the class. You’ll be taken to an edit page of the duplicated class.

1.2. Managing Classes

For each class you can keep track of regular students using reservations, have a waitinglist for reservations, keep track of attendance and set the status of a particular class. In the schedule the status of a class is shown by a colored dot:

- Green - Normal
- Blue - Subteacher
- Red - Open / Sub teacher required
- Orange - Cancelled

5.3.2 2. Studio staff

To add a new staff schedule press the add button in the top-righthand corner. Choose a Location, Shift name (add under School -> Shift), Weekday, Start and End time, Startdate and Enddate. After that you can assign an employee to that shift. To add Employees see under School -> Employee. On the main page you can manage the current shifts for the Studio staff.

5.4 Customers

You can store a lot of information about your customers in OpenStudio.

5.4.1 1. Information

- General information like name, address and comments.
- Subscriptions
- Class cards
- Class attendance
- Class reservations
- Workshop registrations

- Payment information
- Documents
- Tasks (to-do list)
- Invoices

5.4.2 2. Pause A Subscription

To pause a subscription go to the edit page for a customer and then click the subscriptions link and then the “Pause” button for the subscription you wish to pause.

5.5 Workshops

To add a workshop, follow these steps:

Add a workshop Add at least 1 activity to the workshop agenda (Optional) Add a product that links to the activity you just created Note: All activities are automatically linked to the auto-created “Full workshop” product.

5.5.1 1. Manage

1.1. Products

A product is a collection of activities from the agenda. By default a full workshop product is created, which can't be deleted. By adding customers to a product you can keep track of payments and automatically get an overview of expected attendance in the workshop agenda.

1.2. Agenda

The agenda page is used to manage activities for a workshop. You can schedule new activities, manage existing ones and keep track of the attendance for all activities.

5.5.2 2. Tasks

Here you can keep track of things to do or to remember for this workshop. These memos will show up on the pinboard.

5.5.3 3. Quick Stats

This page gives a quick overview of the revenue and which cities most of the customers are from.

5.6 Settings

OpenStudio is configured using the settings pages

5.6.1 1. General

General settings

Separate customers by location In case you have multiple physical locations where you teach, you might want to keep track of which customer is attending classes where. By turning this option on, an extra dropdown box appears in the customers edit pages and collection & payment export pages allowing selection of the location. **Show welcome message** In case you want to turn the welcome message back on, you can do so here. **Currency** This is used in the csv export for collection and payment with customers. Add the 3 letters specifying the currency, eg. EUR, USD, GBP, KRW, etc. **Date format** Choose how dates are displayed.

5.6.2 2. Permissions

Starting with OpenStudio 2.05 a group based permissions model is available in OpenStudio. This model allows you to determine who can see/edit what. It's basic structure is like this: A user is a member of a group. A group has permissions assigned to it which determine what the members of the group can see and edit.

First go to settings → users & groups → groups and add a new group. Once the group is added, you'll see a permissions link for that group in the groups list. By clicking that link you can set which permissions that group has. The next step is to add a user to that group. Go to preferences → users & groups → users and select a user. Then click the group link left of the edit button. In the menu shown now you can select a group to add the user to.

Please note that the group 'admin' always has full access to everything.

5.7 Best Practices

5.7.1 1. Subscriptions

When using the collection exports to collect payments from customers using automated software, make sure only the subscriptions for which the fees have to be collected are listed in the required months. For example when collecting the fees for one subscription a month, make sure there is only one subscription active for each customer. The best way to do this is to change subscriptions at the month boundaries, so the old subscription ends at the last day of the month and the new subscription starts at the first day of the next month. This way there is no overlap between the old and new subscriptions and no duplicate collections occur.

Most customer related administrative activities start at the Customers>List Customers page.

6.1 1. Adding Customers

Click the +Add button at the top right of the Customer List window. In add customer popup, input the First Name, Last Name

The email address will become their customer login username in case you choose to have your customers participate in the OpenStudio system.

6.2 2. Exports

You can export a couple different lists from Customers>List on Customers page by clicking on the small Cloud button near the top right of the window.

You can export a customer list complete with profile data to excel by clicking Active Customers under the Cloud Button at the top right of the screen.

You can export an email list by clicking on 'Mailing List' under the Menu Icon at the top right of the screen.

6.3 4. # of records to view in Customer List

In order to select between 10, 15 or 25 records per page, click on the button with eye symbol.

6.4 5. Search

You can quickly and easily search for a Customer record by typing characters from their name into the Search field at the top left. Results will automatically be displayed in the main body of the form.

6.5 6. Send e-mail

Clicking on the envelop to the right side of a customer in the list will open a new email with the recipient email address already populated in the default email client, if installed.

6.6 7. Edit

You can select the student record you wish to edit by clicking on their picture, or by clicking pencil icon button to the right of the students record.

6.7 8. Delete

If you want to remove a customer from this list, you would click on the X icon Button to the right of the customer record. The customer record will be moved to the 'Deleted' tab.

6.7.1 Customers Record

You can store a lot of information about your customers in OpenStudio.

6.8 1. Information

- General information like name, address and comments.
- Subscriptions
- Class cards
- Class attendance
- Class reservations
- Workshop registrations
- Payment information
- Documents
- Tasks (to-do list)
- Invoices

6.9 2. Pause A Subscription

To pause a subscription go to the edit page for a customer and then click the subscriptions link and then the "Pause" button for the subscription you wish to pause.

Manage subscription credits

This page describes how to manage credits for customers subscriptions. One subscription credit equals access to one class.

7.1 Configuring school subscriptions

The following fields for a school subscription define credit behavior

1. Classes
2. Classes per
3. Reconciliation classes
4. Credits validity (days)

Classes & Classes per Classes define the number of classes a week or month. When classes are defined per week, OpenStudio will automatically calculate the right number of credits for a month when giving out credits. *Example: 1 class a week for a month with 31 days would be 4.4 credits*

Reconciliation classes Like in the example above, some months might have 5 Mondays, so 4.4 credits isn't enough. The field *Reconciliation classes* allows you to define how many negative credits a customer can accumulate on this subscription. Or in other words, how many classes customers can attend on this subscription without having credits. Looking at a whole year, customers will get enough credits, but might have a temporary shortage depending on the number of Mondays, Tuesdays, etc. in a month.

Credits validity (days) Number of days subscription credits will be valid. Think of it like the minutes on your phone subscription, when they're not used within a certain period, they expire and can no longer be used.

7.2 Adding credits for all subscriptions

After setting the fields above for all appropriate subscriptions under school and adding subscriptions to customers, it's possible to add credits to all subscriptions for the selected month in one go.

Go to customers in the main menu and select *Subscription credits*. Select the tab *Added credits*. After choosing the month for which credits should be added, click the *Add* button in the top right and choose *Continue*. This operation might take a few minutes, when there are a few hundred (or more) subscriptions. The operation will continue in the background, you can continue working as usual. After a few minutes you can come back or refresh the page to get a status update.

7.3 Expiring credits

When *Credits validity* is set for one or more school subscriptions, you can expire credits. Go to customers in the main menu and select *Subscription credits*. Select the tab *Expired credits*. Click the *Expire* button in the top right of the page to remove all expired credits from all subscriptions.

7.4 Credit mutations for a single subscription

Go to customers - subscriptions - edit a subscription - go to the credits tab. Here you can add credit mutations (add and subtract) by using the add button.

Manage class access

This page will explain how to manage which subscriptions and class cards have access to which classes. This process is the same for subscriptions and class cards. In the explanation below subscriptions will be used.

8.1 Prerequisites

1. Add at least one class under Schedule → classes
2. Create at least one subscription under School → subscriptions
3. Add a subscription group under School → subscriptions and add the subscription created in the step above to the group. In this example the name “MyGroup” will be assumed.

8.2 Allowing a subscription to a class

1. Go to Schedule → classes, click the *Actions* button and select the *Edit* link under *All classes in series*
2. Click the *Subscriptions* tab
3. Click the *Add* button in the top right corner and select the subscription group created earlier.
4. Check the boxes you want to allow for this subscription
5. Click save

8.3 Access levels

Enrollment Allow customers with a subscription in this group to make reservation from the booking pages in the shop.

Book in advance Allow customers with a subscription in this group to book this class in advance.

Attend Allow customers with a subscription in this group to attend this class. *When Book in advance or Enrollment is set, Attend will be set automatically.*

Teacher payments

OpenStudio supports fixed rate and attendance based payments for teachers. Monthly credit invoices specifying the class payments and travel allowances can be generated for each teacher.

9.1 Fixed rate

Fixed rate payments allow setting a fixed rate for classes in selected school locations. A default rate should be set for each teacher, where class specific rates can be set to override the default rate. Travel allowance is entered by location, if a class is given in a selected location the travel allowance is added. Consecutive classes won't generate travel allowance twice on the invoice. A class is seen as consecutive if the start time of the next class is within 30 minutes of the previous class on the same day.

9.1.1 Overview

Using fixed rate teacher payments has a few steps to set up and components used,

1. Store teacher bank account information
2. Setting rates and travel allowances for teachers
3. Create credit invoices
4. Paying credit invoices by creating a payment batch

9.1.2 Store bank account information

School - Teachers and click the *Actions* button for the selected teacher, choose edit and navigate to the *Payment info* tab. Add the bank account information on this page.

9.1.3 Setting rates

Go to School - Teachers and click the *Actions* button for the selected teacher and choose *Fixed rate payments*. Add a default rate and any class specific rates.

9.2 Attendance based

Attendance based payments allow setting a flexible rate for classes of selected types. Attendance lists are populated, for example 1 student means €1, 2 students means €2, etc. These lists are then linked to classtypes.

9.2.1 Overview

Setting up attendance based payments for teachers has a few steps

1. Setting the teacher payment rate type under setting
2. Store teacher bank account information
3. Setting rates
4. Create credit invoices

9.2.2 Setting the teacher payment rate type

Go to settings in the main menu and choose Financial. Here go to the tab *Teacher payments*. Select *Attendance based* and click *Save*.

9.2.3 Store bank account information

School - Teachers and click the edit button for the selected teacher and navigate to the *Payment info* tab. Add the bank account information on this page.

9.2.4 Setting rates

Go to School - Teachers and click the *Payment Attendance lists* tab. Create a new list by clicking *Add*. Once the list is created, rates can be added by clicking the *Rates* button for the selected list and classtypes can be linked using the *Class types* button. One rate has to be added for each number of students attending a class.

Note: A class type can have only one list associated with it

9.3 Travel allowance

Travel allowance is entered by location, if a class is given in a selected location the travel allowance is added. Consecutive classes won't generate travel allowance twice on the invoice. A class is seen as consecutive if the start time of the next class is within 30 minutes of the previous class on the same day. If OpenStudio finds consecutive classes, travel allowance is only added to the first class of the consecutive classes.

9.3.1 Configuration

Travel allowance is set individually for each teacher & location.

1. Go to School - Teachers in the main menu
2. Click the *Actions* button for the selected teacher and choose *Travel allowance*.
3. Click *Add* to add a new travel allowance for a teacher.

9.4 Create credit invoices

After configuring a payment rate system for teachers, credit invoices can be created.

1. Go to Finance - Teacher payments in the main menu
2. Go to the tab “Not verified”
3. Click Find classes and choose a range of dates to search for any classes not yet found
4. Verify all classes or verify individual classes
5. Go to the tab “Verified”
6. By clicking process credit invoices will be created for all verified classes or verified classes in a chosen range

9.5 Paying credit invoices

The easiest way to pay the credit invoices for teachers is to create a payment batch. Navigate to Finance - Batch payments and click add. Choose Teacher payments when asked what kind of batch to create. Here fill out the form with desired data and click *Save*.

The batch created will contain all teacher payment credit invoices with status *Sent* for the selected month.

In the batch view, click the *Export* button to export the batch. OpenStudio exports the batch items in a .csv file. This file can be transformed to different formats. For example to the PAIN format by using an application called IBANC.

Set the batch status to *Sent to Bank* when the batch has been sent to the bank. This will automatically add payments to all invoices in the batch and set the invoice statuses to *paid*.

10.1 Summary

Customers can manage their subscription to your MailChimp mailing lists from their personal profile using this integration. A customer is shown as subscribed based on whether the email address they used to log on to OpenStudio is found in the MailChimp mailing list. Please be aware that it's not possible to manually subscribe customers to mailing lists, as in many parts of the world it's now required that customers opt-in for mailing lists themselves.

In case there is a demand for this feature, please let us know by creating an issue on our github page or commenting on it when an issue has already been created.

10.2 Prerequisites

Before setting up MailChimp integration in OpenStudio, make sure you have the following things ready. In case you can't find them, refer to the MailChimp knowledgebase or contact MailChimp support.

1. Your MailChimp username
2. Your MailChimp API key
3. The MailChimp *list_id* of any lists you would like your customers to be able to sign up for from OpenStudio

10.3 Setup

Entering the username and API key

1. Log in with an user account having admin privileges in OpenStudio
2. Go to settings - Integration - MailChimp
3. Enter your username and API key and Save

Linking OpenStudio lists to your MailChimp mailing lists

1. Go to Settings - Mail - Mailing lists
2. Add a mailing list (the name doesn't have to match the MailChimp name) and link the list to MailChimp by setting the desired MailChimp *list_id*. The *list_id* is what determines which name in OpenStudio is matched with which list in MailChimp.
3. When based in the EU, make sure you have set a clear description of your mailing list and defined the frequency (eg. once a month).
4. Save

Enable Mailing lists in customer profiles

1. Go to settings - Shop - Customer profiles
2. Check the *Mailing lists* checkbox
3. Save

11.1 Summary

The Exact Online integration is a one way system. It assumes all data originates from OpenStudio. The following information can be transferred from OpenStudio to Exact Online:

- Invoices
- Invoice items
- Customers
- Customers Bank accounts
- Customers Bank account mandates

11.2 Prerequisites

Before setting up Exact Online integration in OpenStudio, make sure you have the following things ready. In case you can't find them, refer to the Exact Online knowledgebase or contact Exact support. Please keep in mind that the API urls change depending on your region. The client ID and secret are obtained using the Exact Online App center. When registering an app Exact Online will ask for a Redirect URI. This is the base URL of your OpenStudio installation, followed by `/exact_online/oauth2_success`. Also make sure the url starts with https, as Exact Online doesn't accept insecure redirect URI's. eg. https://demo.openstudioproject.com/exact_online/oauth2_success.

1. Exact Online Server auth URL
2. Exact Online Server rest URL
3. Exact Online Server token URL
4. Client Base URL (The base URL of your OpenStudio installation)
5. Exact Online API Client ID
6. Exact Online API Client Secret

11.3 Setup

OpenStudio settings

Complete these steps before actually linking OpenStudio to Exact Online

1. Set the GLAaccount field for all OpenStudio sellables. Classes (prices), event tickets, class cards, subscriptions and memberships. The values of these fields should correspond to the correct General Ledger Account numbers of Exact Online.
2. Under Settings - Financial - Invoices - Groups, set the appropriate journal ID.
3. Under Settings - Financial - Tax rates, edit each tax rate and set the appropriate Vat Code ID, as found in Exact Online.
4. Under settings - Financial - Payment methods - edit each payment method and set the Accounting code to the payment methods ID as found in Exact Online.

Establish link

1. Log in with an user account having admin privileges in OpenStudio
2. Go to settings - Integration - Exact Online
3. Enter data prepared in Prerequisites and click save
4. Click the green *Authorize* button and follow the steps until redirected back to OpenStudio
5. Choose a default Division and click back

The link between Exact Online and OpenStudio should be established.

12.1 About

- All returned data is encoded using UTF-8
- Weeks start on Monday
- All calls to can be made as JSON and XML

12.2 schedule_get

The schedule API is a one way API that allows you to get information from the class schedule in OpenStudio.

12.2.1 1. Call

Calls can be made as JSON and as XML. Both return the same data, just formatted according to the call you make.

json `https://<hosting location>/api/schedule_get.json`

xml `https://<hosting location>/api/schedule_get.json`

Mandatory variables

user API user

key Key for API user

week ISO week number of year, 1 - 52 (or 53 if applicable)

year year as YYYY

Optional variables

TeacherID If specified, only returns classes containing the specified teacher. This works for regular and substitute teachers

ClassTypeID If specified, only returns classes of the specified type

LocationID If specified, only returns classes for the specified location

LevelID If specified, only return classes with the specified level

SortBy

Accepted values: 'location' and 'time'

- **location** Sort by location and then time
- **time** Sort by time and then location

Example calls

```
https://<hosting location>/api/schedule_get.json?user=test&key=test&week=1&
↪year=2014
https://<hosting location>/api/schedule_get.xml?user=test&key=test&week=1&
↪year=2014&TeacherID=1&ClassTypeID=1
```

12.2.2 2. Return

The global structure for the returned values is as follows. The actual values returned differ slightly depending on whether called as XML or JSON.

data (top level) classtypes¹

teachers²

locations³

classes⁴

Monday - Sunday date (yyyy-mm-dd)

1. The following data is provided for classtypes

Description [String] Description of classtype

Id [String] ID of classtype

Link [String] URL to classtype page on website (optional)

LinkThumbLarge [String] URL to large thumbnail for class (400px*400px)

LinkThumbSmall [String] URL to small thumbnail for class (50px*50px)

Name [String] Name of classtype

2. The following data is provided for teachers

Bio [String] Biography of teacher

Id [String] ID of teacher

LinkToBio [String] URL to teachers' online Biography

LinkThumbLarge [String] URL to teacher picture large thumbnail

Name [String] Name of teacher

3. The following data is provided for locations

Id [String] ID of location

Name [String] Name of location

4. The following data is provided for a class

BookingOpen [Date] Date from which bookings for this class will be accepted (YYYY-MM-DD)

BookingStatus [String] Booking status

BookingSpacesAvailable [String] Available spaces for online booking

Cancelled [Boolean] True if the class has been cancelled False when not

CancelledDescription [String] Description of why the class is cancelled (If entered)

ClassTypeID [String] ID of classtype

ClassType [String] Name of classtype

CountAttendance [String] Number of students attending (having booked)

CountReservations [String] Number of reservations

CountReservationsCancelled [String] Number of cancelled reservations

Endtime [String] End time of class

Holiday [Boolean] True when a holiday is found in OpenStudio for the location of this class False when not

HolidayDescription [String] Description of holiday

LevelID [String] ID of class level

Level [String] Name of class level

LinkShop [String] URL to class in OpenStudio shop

LocationID [String] ID of location

Location [String] Name of location

MaxStudents [String] Max. spaces in this class

Starttime [String] Start time of class

Subteacher [Boolean] True if the current teacher or second teacher is a substitute teacher False when not

Teacher [String] Name of teacher (Firstname lastname)

Teacher2 [String] Name of second teacher (Firstname lastname)

TeacherID [String] ID of teacher

TeacherID2 [String] ID of second teacher

12.3 schedule_get_days

The schedule API is a one way API that allows you to get information from the schedule in OpenStudio. This chapter described the schedule_get_days endpoint which allows you get get schedule information in a selected range of dates.

12.3.1 1. Call

Calls can be made as JSON and as XML. Both return the same data, just formatted according to the call you make.

JSON https://<hosting location>/api/schedule_get_days.json

XML https://<hosting location>/api/schedule_get_days.xml

1.1. Mandatory Variables

user API user

key Key for API user

date_start Start of day range as yyyy-mm-dd

date_end End of day range as yyyy-mm-dd

1.2. Optional Variables

TeacherID If specified, only returns classes containing the specified teacher. This works for regular and substitute teachers

ClassTypeID If specified, only returns classes of the specified type

LocationID If specified, only returns classes for the specified location

LevelID If specified, only return classes with the specified level

SortBy Accepted values: 'location' or 'time'

location: Sort by location and then time

time: Sort by time and then location

1.3. Example Calls

```
https://api/schedule_get_days.xml?user=test&key=test&date_start=2016-01-01&
↪date_end=2016-01-06
```

```
https://api/schedule_get_days.xml?user=test&key=test&date_start=2016-01-01&
↪date_end=2016-01-06&TeacherID=1&ClassTypeID=1
```

12.3.2 2. Return

The global structure for the returned values is as follows. The actual values returned differ slightly depending on whether called as XML or JSON.

data (top level) (dict)

- schedule (list)** date (yyyy-mm-dd)
- classes ¹
- classtypes (list) ²
- teachers (list) ³
- locations (list) ⁴
- levels (list) ⁵

1. The following data is provided for a class

- BookingOpen** [Date] Date from which bookings for this class will be accepted (YYYY-MM-DD)
- BookingStatus** [String] Booking status
- BookingSpacesAvailable** [String] Available spaces for online booking
- Cancelled** [Boolean] True if the class has been cancelled False when not
- CancelledDescription** [String] Description of why the class is cancelled (If entered)
- ClassTypeID** [String] ID of classtype
- ClassType** [String] Name of classtype
- CountAttendance** [String] Number of students attending (having booked)
- CountReservations** [String] Number of reservations
- CountReservationsCancelled** [String] Number of cancelled reservations
- Endtime** [String] End time of class
- Holiday** [Boolean] True when a holiday is found in OpenStudio for the location of this class False when not
- HolidayDescription** [String] Description of holiday
- LevelID** [String] [String] ID of class level
- Level** [String] Name of class level
- LinkShop** [String] URL to class in OpenStudio shop
- LocationID** [String] ID of location
- Location** [String] Name of location
- MaxStudents** [String] Max. spaces in this class
- Starttime** [String] Start time of class
- Subteacher** [Boolean] True if the current teacher or second teacher is a substitute teacher False when not
- TeacherID** [String] ID of teacher
- TeacherID2** [String] ID of second teacher

Teacher [String] Name of teacher (Firstname lastname)

Teacher2 [String] Name of second teacher (Firstname lastname)

2. The following data is provided for classtypes

Description [String] Description of classtype

Id [String] ID of classtype

Link [String] URL to classtype page on website (optional)

LinkThumbLarge [String] URL to larg thumbnail for class (400px*400px)

LinkThumbSmall [String] URL to small thumbnail for class (50px*50px)

Name [String] Name of classtype

3. The following data is provided for teachers

Bio [String] Biography of teacher

Id [String] ID of teacher

LinkToBio [String] URL to teachers' online Biography

LinkThumbLarge [String] URL to teacher picture large thumbnail

Name [String] Name of teacher

4. The following data is provided for locations

Id [String] ID of location

Name [String] Name of location

5. the following data is provided for levels

Id [String] ID of level

Name [String] Name of level

12.4 workshops_get

The workshops API is a one way API that allows you to get information from OpenStudio about upcoming workshops.

12.4.1 1. Call

Calls can be made as JSON and as XML. Both return the same data, just formatted according to the call you make.

JSON https://<hosting location>/api/schedule_get.json

XML https://<hosting location>/api/schedule_get.xml

1.1 Mandatory Variables

user API user

key Key for API user

1.2 Example Calls

```
https://<hosting location>/api/workshops_get.json?user=test&key=test
```

12.4.2 2. Return

The global structure for the returned values is as follows. The actual values returned differ slightly depending on whether called as XML or JSON.

data (top level)(list) workshops (list) ¹

1. The following data is provided for a workshop

Description [String] Workshop description

Enddate [Date] Workshop end Date

Endtime [String] End time of class

id [BigInt] ID of workshop

LinkImage [String] URL to uploaded picture

LinkThumbLarge [String] URL to large thumbnail for workshop (400px * 400px)

LinkThumbSmall [String] URL to small thumbnail for workshop (50px * 50px)

LinkShop [String] URL to OpenStudio shop

Location [String] Name of location

LocationID [BigInt] ID of location

Name [String] Workshop Name

Preview [String] Workshop short description

Price [Float] Workshop price (for full workshop product)

Startdate [Date] Workshop start Date

Starttime [String] Start time of class

Tagline [String] Workshop Tagline

Teacher ² [Dict] Dict with teacher info

Teacher2 ³ [Dict] Dict with teacher2 info

2. Teacher fields

Bio [String] Teacher Biography

id [BigInt] ID of teacher

LinkToBio [String] URL to teacher biography

LinkThumbLarge [String] URL to large thumbnail for workshop (400px * 400px)

LinkThumbSmall [String] URL to small thumbnail for workshop (50px * 50px)

Name [String] Teacher Name

Role [String] Teacher Role

Website [String] URL to teacher Website

12.5 workshop_get

The workshops API is a one way API that allows getting information from OpenStudio about a single workshop. Please note that the workshop has to be marked as visible in the shop in OpenStudio.

12.5.1 1. Call

Calls can be made as JSON and as XML. Both return the same data, just formatted according to the call you

JSON https://<hosting location>/api/workshop_get.json

XML https://<hosting location>/api/workshop_get.xml

1.1. Mandatory Variables

user API user

key Key for API user

id Workshop ID

1.2. Example Calls

```
https://api/schedule_get_days.xml?user=test&key=test&date_start=2016-01-01&
↳date_end=2016-01-06
```

```
https://api/schedule_get_days.xml?user=test&key=test&date_start=2016-01-01&
↳date_end=2016-01-06&TeacherID=1&ClassTypeID=1
```

12.5.2 2. Return

The global structure for the returned values is as follows. The actual values returned differ slightly depending on whether called as XML or JSON.

data (top level) (list) workshop ¹

1. The following data is provided for a workshop

Activities³ [List] List of activities for workshop, sorted by date and then time

Cancelled [Boolean] True if the class has been cancelled False when not

Description [String] Workshop Description

Enddate [Date] Workshop end Date

Endtime [String] End time of class

Holiday [Boolean] True when a holiday is found in OpenStudio for the location of this class False when not

id [BigInt] ID of workshop

LinkImage [String] URL to uploaded picture

LinkThumbLarge [String] URL to large thumbnail for workshop (400px * 400px)

LinkThumbSmall [String] URL to small thumbnail for workshop (50px * 50px)

LinkShop [String] URL to OpenStudio shop

Location [String] Name of location

LocationID [BigInt] ID of location

Name [String] Workshop Name

Preview [String] Workshop Preview

Price [Float] Workshop price (for full workshop product)

Startdate [Date] Workshop start Date

Starttime [String] Start time of class

Subteacher [Boolean] True if the current teacher or second teacher is a substitute teacher False when not

Tagline [String] Workshop Tagline

Teacher² [Dict] Dict with teacher info

Teacher2² [Dict] Dict with teacher2 info

2. Teacher fields

Bio [String] Teacher Biography

id [BigInt] ID of teacher

LinkToBio [String] URL to teacher biography

LinkThumbLarge [String] URL to large thumbnail for workshop (400px * 400px)

LinkThumbSmall [String] URL to small thumbnail for workshop (50px * 50px)

Name [String] Teacher Name

Role [String] Teacher Role

Website [String] URL to teacher Website

3. Activity

Date [Date] Date of activity
Endtime [String] End time (HH:MM)
id [BigInt] ID of activity
Location [String] Name of location
LocationID [BigInt] ID of location
Name [String] Name of activity
Starttime [String] Start time (HH:MM)
Teacher [String] Name of teacher
Teacher2 [String] Name of teacher2
TeacherID [BigInt] ID of teacher
TeacherID2 [BigInt] ID of teacher2

12.6 school_subscriptions_get

Returns information from school - subscriptions

12.6.1 1. Call

Calls can be made as JSON and as XML. Both return the same data, just formatted according to the call you make.

JSON https://<hosting location>/api/schedule_get.json

XML https://<hosting location>/api/schedule_get.xml

1.1 Mandatory Variables

user API user

key Key for API user

1.2 Example Calls

```
https://<hosting location>/api/school_subscriptions_get.json?user=test&key=test
```

12.6.2 2. Return

The global structure for the returned values is as follows. The actual values returned differ slightly depending on whether called as XML or JSON.

data (top level)(list) Subscriptions ¹

1. The following data is provided for a subscription

Classes [Int] Number of Classes

Description [String] Subscription Description

LinkShop [String] URL to subscription page in OpenStudio shop

Name [String] Subscription Name

Price [Float] Subscription Price

SortOrder [int] Subscription sorting order

SubscriptionUnit [String] 'week' or 'month' / unit to define number of Classes

Unlimited [Boolean] True if unlimited classes, else False

12.7 school_classcards_get

Returns information from school - subscriptions

12.7.1 1. Call

Calls can be made as JSON and as XML. Both return the same data, just formatted according to the call you make.

JSON https://<hosting location>/api/school_classcards_get.json

XML https://<hosting location>/api/school_classcards_get.xml

1.1 Mandatory Variables

user API user

key Key for API user

1.2 Example Calls

```
https://<hosting location>/api/school_classcards_get.json?user=test&key=test
```

12.7.2 2. Return

The global structure for the returned values is as follows. The actual values returned differ slightly depending on whether called as XML or JSON.

data (top level) (list) Class card1 ¹

1. The following data is provided for a class card

Classes [Int] Number of classes

Description [String] Card description

Name [String] Card name

Price [Float] Card price

TrialCard [Boolean] True if this is a trial card, else False

Unlimited [Boolean] True if unlimited classes, else False

Validity [Int] Validity number

ValidityUnit [String] 'day', 'week' or 'month'

The right of consent is implemented in the following ways:

1. Customers must accept your privacy notice before being able to register (when a privacy notice has been set)
2. Acceptance of privacy notices is logged
3. Customers cannot be automatically added to mailing lists on registration. Please refer to the MailChimp integration in this manual for more information

13.1 Setting a privacy notice

As a business it's your responsibility to have a clear data processing plan and convey that to any customer when they start using your services. As OpenStudio can be used and implemented in many different scenarios, we don't provide standard privacy notices. For examples of good and bad practices when formulating a privacy notice, please have a look at the website of the [ico](#).

A privacy notice is linked to an organisation in OpenStudio. A link to the privacy notice of the Default organisation will be shown on the registration screen for customers.

1. Go to Settings - System - Organisations
2. Add an organisation if no organisation is added yet or edit the default organisation
3. Use the fields *Link to privacy notice* and *Privacy notice version* to link to the privacy notice on your website and set the version that is logged when a customer accepts the document.
4. Save

13.2 View logged Acceptance

The acceptance of privacy notices is recorded in a customers' account.

1. Go to Customers in the main menu and edit a customer

2. Go to the Account tab and select *Accepted documents* - here will be a list of documents this customer has accepted through OpenStudio.

CHAPTER 14

Right to access

Under the GDPR organizations processing personal data are required to provide an electronic copy in a reasonable format to the owners of the data. In OpenStudio customers can view and download all of their data and all data linked to their account from their profile. To enable this feature;

1. Go to Settings - Shop - Customer profiles
2. Make sure the *Privacy* box is checked
3. Save

When the privacy box is enabled, there will be a *Privacy* link under the customer ID in the profile page in the account for customers. This enabled your customers to download their own data at any time, minimizing administrative burden on your organization.

Right to be forgotten

Under the new regulations businesses based in the EU no longer have the right to keep processing any personal data that is no longer relevant to original purposes for which the owner of the data gave his or her consent.

As a general assumption in the case of OpenStudio, personal data no longer relevant can be defined as data of customers who have had no activity in or dealings with your businesses after a given date. To assist in cleaning up inactive customers, a new report has been added. Using this report you can first view a list of customers without activity after a given date and then continue to remove them.

Delete a all customers inactive after a given date

1. Go to Reports - Inactive customers
2. Select date after which OpenStudio is to check for customer activity
3. Run report
4. When ready to delete customer data, click the red “Delete customers” button

Delete a single customer

1. Go to Customers
2. Click the delete button next the the customer to be deleted
3. To to deleted
4. Click the delete button next the the customer to be deleted

Note: All invoices generated by services any customer has used will be retained to comply with financial legislation.

Activity is defined as follows

1. Having a subscription after date provided
2. Having a class card after date provided
3. Having attended an event after date provided
4. Having attended a class
5. Having placed an order

6. Having an invoice with an invoice date after date provided
7. Having logged in
8. Having been created or having registered after date provided
9. Employees having added a note after date provided
10. Being a teacher
11. Being an employee

CHAPTER 16

License & Disclaimer

OpenStudio is licenced under the GPL version 2 or later.

Disclaimer Of Warranty

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF QUALITY, PERFORMANCE, NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. FURTHER, EDWIN VAN DE VEN DOES NOT WARRANT THAT THE SOFTWARE OR ANY RELATED SERVICE WILL ALWAYS BE AVAILABLE.

Limitations Of Liability

YOU ASSUME ALL RISK ASSOCIATED WITH THE INSTALLATION AND USE OF THE SOFTWARE. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS OF THE SOFTWARE BE LIABLE FOR CLAIMS, DAMAGES OR OTHER LIABILITY ARISING FROM, OUT OF, OR IN CONNECTION WITH THE SOFTWARE. USERS OF THE SOFTWARE ARE SOLELY RESPONSIBLE FOR DETERMINING THE APPROPRIATENESS OF USE AND ASSUME ALL RISKS ASSOCIATED WITH ITS USE, INCLUDING BUT NOT LIMITED TO THE RISKS OF PROGRAM ERRORS, DAMAGE TO EQUIPMENT, LOSS OF DATA OR SOFTWARE PROGRAMS, OR UNAVAILABILITY OR INTERRUPTION OF OPERATIONS.

More coming soon!