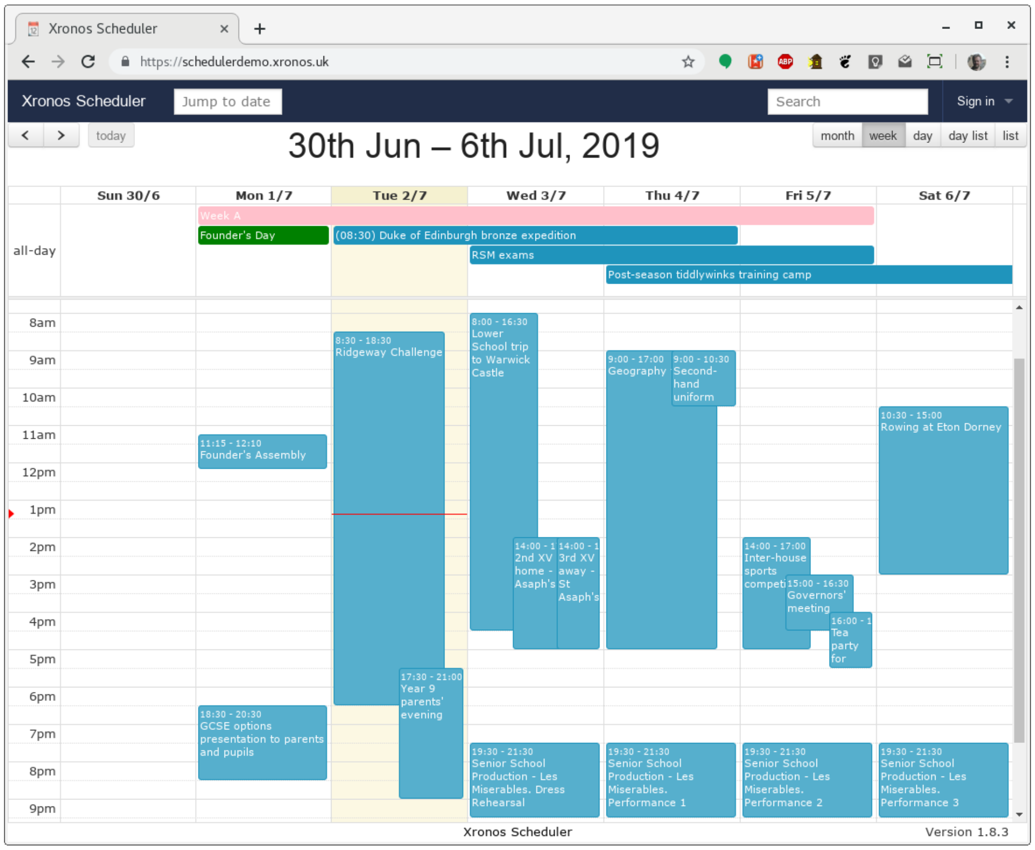
Xronos Scheduler Admin Guide

Release 1.17.2

John Winters

Oct 15, 2024

CONTENTS


Xronos Scheduler
https://schedulerdemo.xronos.uk
Xronos Scheduler
Jump to date
Search
Sign in
today
30th Jun – 6th Jul, 2019
month
week
day
day list
list
Sun 30/6
Mon 1/7
Tue 2/7
Wed 3/7
Thu 4/7
Fri 5/7
Sat 6/7
all-day
Week A
Founder's Day
(08:30) Duke of Edinburgh bronze expedition
RSM exams
Post-season tiddlywinks training camp
8am
9am
10am
11am
12pm
1pm
2pm
3pm
4pm
5pm
6pm
7pm
8pm
9pm
11:15 - 12:10 Founder's Assembly
18:30 - 20:30 GCSE options presentation to parents and pupils
8:30 - 18:30 Ridgeway Challenge
17:30 - 21:00 Year 9 parents' evening
8:00 - 16:30 Lower School trip to Warwick Castle
14:00 - 1 2nd XV home - Asaph's
14:00 - 1 3rd XV away - St Asaph's
19:30 - 21:30 Senior School Production - Les Miserables. Dress Rehearsal
9:00 - 17:00 Geography
9:00 - 10:30 Second-hand uniform
19:30 - 21:30 Senior School Production - Les Miserables. Performance 1
14:00 - 17:00 Inter-house sports competi
15:00 - 16:30 Governors' meeting
16:00 - 1 Tea party for
19:30 - 21:30 Senior School Production - Les Miserables. Performance 2
10:30 - 15:00 Rowing at Eton Dorney
19:30 - 21:30 Senior School Production - Les Miserables. Performance 3
Xronos Scheduler
Version 1.8.3

Xronos Scheduler - Admin Guide

This guide is intended for those responsible for administering an installation of Xronos Scheduler. It contains sections dedicated to various tasks.

It is currently very incomplete - expect a lot of extra sections to appear.

Download this guide as a PDF

CHAPTER

ONE

USERS

1.1 Authentication and Authorisation

Scheduler makes a distinction between authentication (identifying who any given person is) and authorisation (granting permission to do certain things).

Authentication is handled by a plug in module called OmniAuth and makes use of external authentication services. It is thus not generally necessary to create user accounts manually within Scheduler. By linking OmniAuth to an existing authentication mechanism used by the school (typically Google) users are set up automatically and do not need to be given a new set of credentials.

Once a user has authenticated, Scheduler then applies various rules to set up initial permissions. The system administrator can tweak the initial settings, and adjust individual users' permissions as desired.

1.2 User profiles

The system by default has three pre-configured user profiles. These are:

- Staff
- Pupil
- Guest

At first installation, the Staff user profile gets quite a few permissions, the Pupil user profile gets very limited ones, and the Guest profile gets none at all. The system administrator can alter the permissions for each of these profiles, and create more profiles if desired.

Each profile carries a flag called "known". By default, staff and pupil profiles have the known flag set to true, whilst the guest profile has it set to false.

Any user who is not "known" has exactly the same access to the system as someone who is not logged in at all. It is thus acceptable for any Google user to authenticate into the system, but they will gain no authorisation by doing it. Only known users get into the authorisation process.

If it is desired to lock a given set of users out of the system - e.g. ex staff - then an additional user profile can be created with the "known" flag set to false. By setting a user to have this profile the user effectively becomes a guest. If at a later date the user is to be re-admitted then the user's profile can simply be changed back.

1.3 First login

When a new user first authenticates in Scheduler, a User record is created and assigned to the "Guest" user profile. The system then does some checks to see whether any more than very basic permissions can be granted.

The e-mail address of the new user is compared against the e-mail addresses of all known staff. If it matches one of them then the user record is linked to that staff record, and the user's profile is changed to the staff profile.

If that fails, then the same process is carried out against all known pupils. Again, if a match is found the user record is linked to the pupil record and the user's profile is changed to the pupil profile.

If both these checks fail, then the new user is left as a guest user but the same checks will be performed again each time the user logs in in the future. This is to cope with the case where a new member of staff is added to the authentication mechanism before being added to the list of staff. At first login, the user will be a guest, but once the necessary staff record is created then the next login will cause permissions to be allocated.

1.4 Manual setup

It is possible that the system administrator will want to grant permissions to someone who is neither a member of staff nor a pupil. E.g. a representative of the PTA.

The extra user will still need to authenticate, and so will need an account on the authentication system. If you are using Google, then a Google e-mail address will suffice.

The user can then be granted permissions by creating a suitable user profile, setting it to be a "known" one, setting the permissions as desired and then editing the user record to link it to that new user profile.

At that point, the user becomes a "known" one and whatever permissions have been granted will be effective.

Once the user has got a user profile other than "Guest user" any subsequent changes are under the control of the system administrator. The automatic process described in the previous section is applied only to guest users.

1.5 Permission flags

Each user profile has a set of permission flags, each of which can be on or off.

Each user record then has the same set of flags, but with the added possibility that a flag can have no value at all.

When a user record is first created then all its permission flags are unset, meaning that all the user's permissions are defined by the user's profile.

The usual way of changing user permissions is by editing the appropriate user profile and turning the flags on or off. This then affects all users with that profile.

Finer grained control can be achieved by editing the user's individual record, where the three choices are "on", "off" and "unset".

The available flags are:

editor Can create events within the system.

repeating events Can create repeating events.

add resources Can add resources dynamically to an event. Without this permission bit, any events which the user creates will get only the resources configured by the system administrator - typically the corresponding pupil.

add notes Can attach notes to events within the system. It is not necessary to be the owner of an event in order to attach notes to it.

edit all A user with this privilege bit can edit any event within the system. Typically granted only to a system adminstrator.

subedit all This is a lesser permission - the user can change aspects of any event within the system, adding and removing resources, but can't change the fundamentals of the event.

privileged The user can make use of the privileged event categories. These are things like "Date - crucial" which if selected will cause an event to show through on everyone's schedule all the time.

groups The user can create groups - collections of other things within the system. E.g. a list of members of a football team.

public In conjunction with the previous flag, this one grants the ability to make the groups public - that is, visible to all users of the system.

forms Can create and edit forms within the system. Forms can be attached to resources to gather in more information about what is needed. For instance, if someone requests catering for their event, a form can be used to collect more information about exactly what is needed.

find free The user can access Scheduler's "Find free" facility, which allows the retrieval of information about who or what is free at a given time. E.g. a free meeting room, or a free prefect.

adjust view This flag allows the user to select what schedules he or she wants to look at. Without this permission bit a user can view only the schedules pre-configured by the system administrator. Typically a pupil would not have this bit set, and thus would be able to look at his or her own timetable, plus public calendars. A staff member would have it set, and could thus look at the timetable of any pupil or staff member.

can roam If set, allow the user to browse through linked records within the system. When looking at a group for instance, the user can click on any pupil within the group and get more information about that pupil, including all the other groups of which he or she is a member.

can su Short for "Can set user". A very powerful permission bit. If this is set then the user can choose to become effectively any other user in the system. Generally set only for system administrators.

exams Can access the functionality specifically related to setting up invigilation slots for exams.

relocate lessons Any staff member can use Scheduler's relocation facilities to move one of his or her own lessons to a different free room - e.g. to an ICT room.

A user with this bit set can do the same for anyone's lessons, and can choose to move a lesson to a room which is not apparently free - e.g. because two lessons are being merged due to most of the pupils being absent.

view forms When an event has a form attached to it, e.g. specifying catering requirements, the owner of the event and the owner of the catering resource can view the contents of the form. In addition, any user with this flag set can view all such forms.

admin If set then the user gains full administrative privileges over the system.

view unconfirmed Some resources within the system require approval to be attached to events. For instance, most users cannot simply put an event into the school's public calendar. When a user tries to put an event in the calendar, the connection between the two starts in an "unconfirmed" state, and until the calendar administrator confirms it, the event will not appear in the calendar to most people. Only the requester and the administrator can see it, and then greyed out.

Users with this permission bit set will see it to, but still greyed out.

edit members When a user is editing a group, the normal interface simply allows elements to be added to or removed from the group - effective as at the date of editing.

Scheduler's groups are however a lot more sophisticated than that - they store the entire chronology of a group's membership. Scheduler can keep track of who was a member of a group on any given date.

Users with this privilege bit can edit the underlying membership records and set start and end dates directly. It is thus possible to set a specific duration for a membership.

can use API Users with this bit set can make use of Scheduler's Application Programming Interface, or API.

can upload files The user can upload files for storage within Scheduler. Such files can be attached to events for download by others.

journals Scheduler keeps a journal of every manual edit to each event. Users with this bit set can view these journals.

CHAPTER

TWO

CONTROLLING RESOURCES

2.1 Concepts

The working of Scheduler revolves around the concept of resources attached to events. The whole package exists simply to record when any individual resource is busy and when it is free.

These resources, can be people, places, or abstract things like services or properties.

In a typical school, many resources are available on a first come, first served basis. The first teacher to schedule an event in a given classroom gets it, and then to anyone coming along later it is busy. No authorization is needed.

For other resources though, a degree of control is needed. Resources which need control could be things like:

- The theatre
- The main public calendar
- The sport calendar
- The sports hall
- Audio-visual support
- Catering
- Cleaning

or anything else which the system administrator chooses.

The theatre may well have an administrator who handles bookings and schedules resources around those bookings.

The main public calendar will certainly have at least one person controlling it - people submit proposed events and then the controller decides which ones can actually appear in the calendar. Likewise for the sport calendar, although it may well have a different controller.

Audio-visual support may be provided by a particular individual. He or she will need to look at requests and decided which ones can be fulfilled.

Catering and cleaning are potentially much more complex. They will probably need associated forms (also provided within Scheduler) and perhaps account codes for the service to be charged to.

Some things can cope with only one booking at a time - the theatre for instance - whilst others may be able to cope with several - e.g. catering. It’s up to the resource administrator to decide what is feasible.

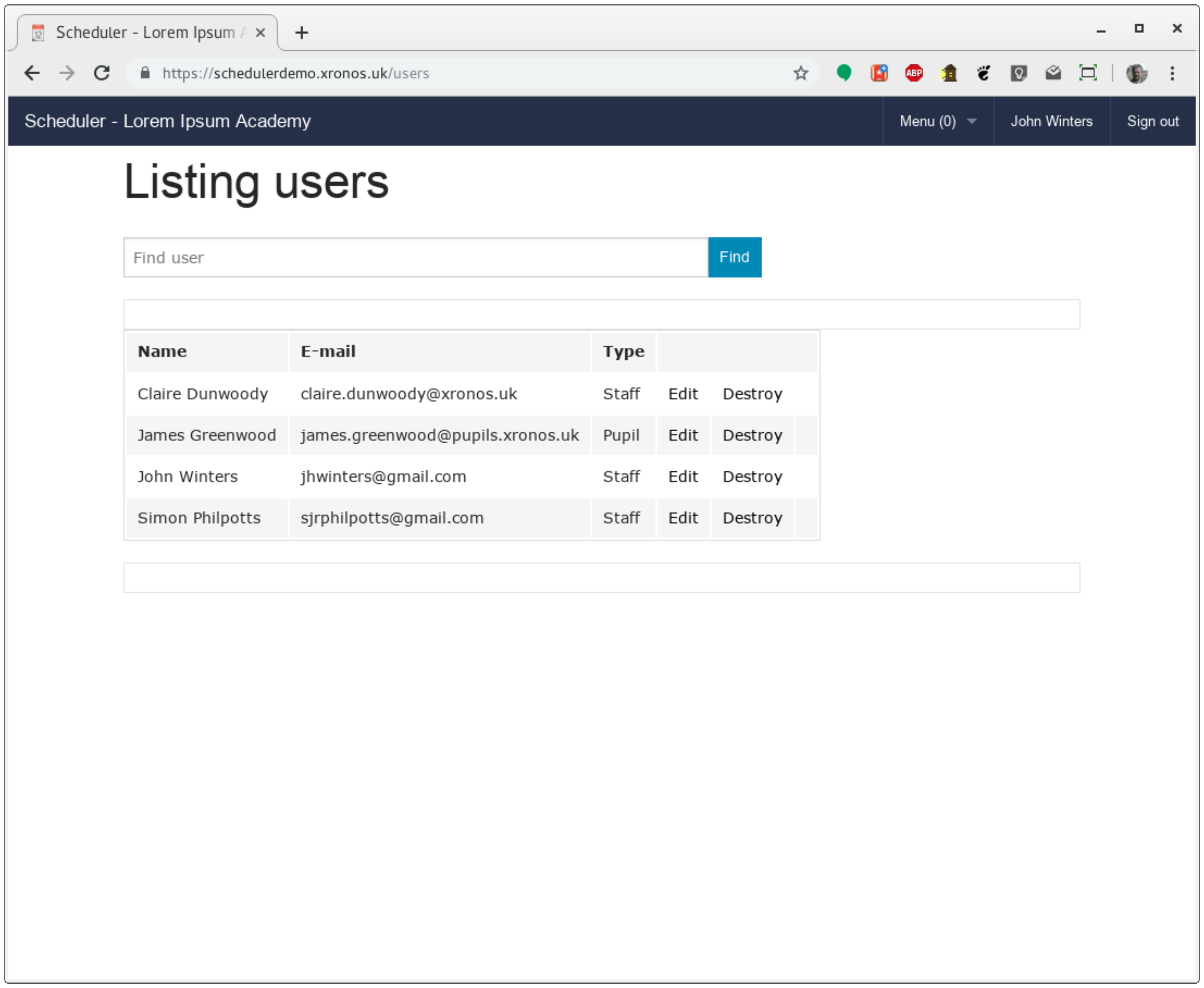
There exist also pooled resources - those where the requirement is for one (or more) resources from the pool, but it doesn’t actually matter to the requester which one is allocated - e.g. minibuses Scheduler handles this case through the use of Resource Groups.

2.2 Adding control

The default for any resource in Scheduler is that it is freely assignable to events. Anyone with edit permission can add any resource to any event which they can edit.

To put a resource under control, one simply needs to allocate one or more controllers to it. This is done by way of the User editing dialogues.

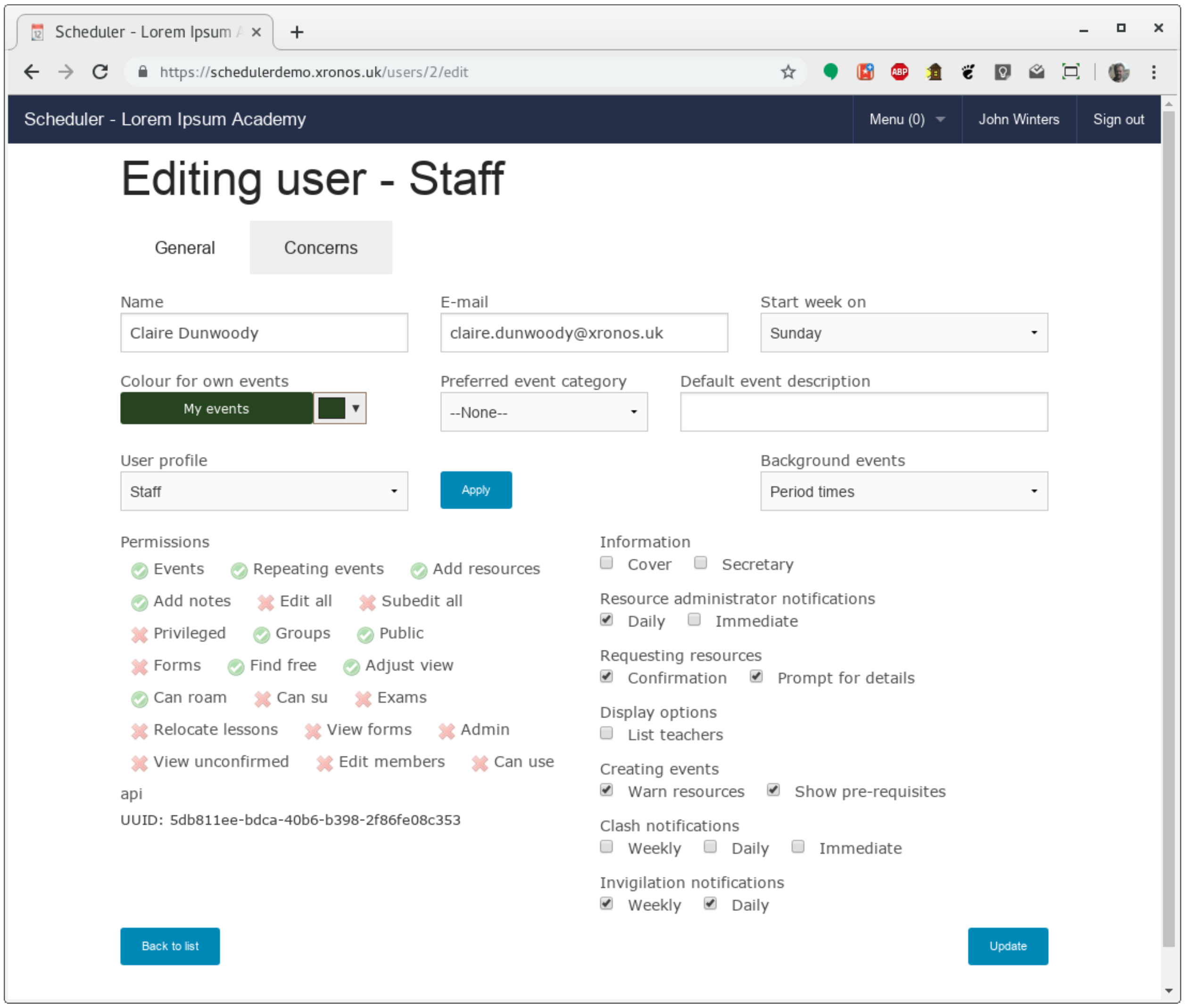
Menu => Admin => Models => Users


Scheduler - Lorem Ipsum
https://schedulerdemo.xronos.uk/users
Scheduler - Lorem Ipsum Academy
Menu (0)
John Winters
Sign out
Find user
Find

Listing users

Name	E-mail	Type		
Claire Dunwoody	claire.dunwoody@xronos.uk	Staff	Edit	Destroy
James Greenwood	james.greenwood@pupils.xronos.uk	Pupil	Edit	Destroy
John Winters	jhwinters@gmail.com	Staff	Edit	Destroy
Simon Philpotts	sjrphilpotts@gmail.com	Staff	Edit	Destroy

Clicking "Edit" for the user Claire Dunwoody produces the following:


Scheduler - Lorem Ipsum A
https://schedulerdemo.xronos.uk/users/2/edit
Scheduler - Lorem Ipsum Academy
Menu (0)
John Winters
Sign out

Editing user - Staff

General | Concerns

Name: Claire Dunwoody

E-mail: claire.dunwoody@xronos.uk

Start week on: Sunday

Colour for own events: My events

Preferred event category: --None--

Default event description:

User profile: Staff [Apply]

Background events: Period times

Permissions

✓ Events ✓ Repeating events ✓ Add resources
✓ Add notes ✗ Edit all ✗ Subedit all
✗ Privileged ✓ Groups ✓ Public
✗ Forms ✓ Find free ✓ Adjust view
✓ Can roam ✗ Can su ✗ Exams
✗ Relocate lessons ✗ View forms ✗ Admin
✗ View unconfirmed ✗ Edit members ✗ Can use

api
UUID: 5db811ee-bdca-40b6-b398-2f86fe08c353

Information
☐ Cover ☐ Secretary

Resource administrator notifications
☑ Daily ☐ Immediate

Requesting resources
☑ Confirmation ☑ Prompt for details

Display options
☐ List teachers

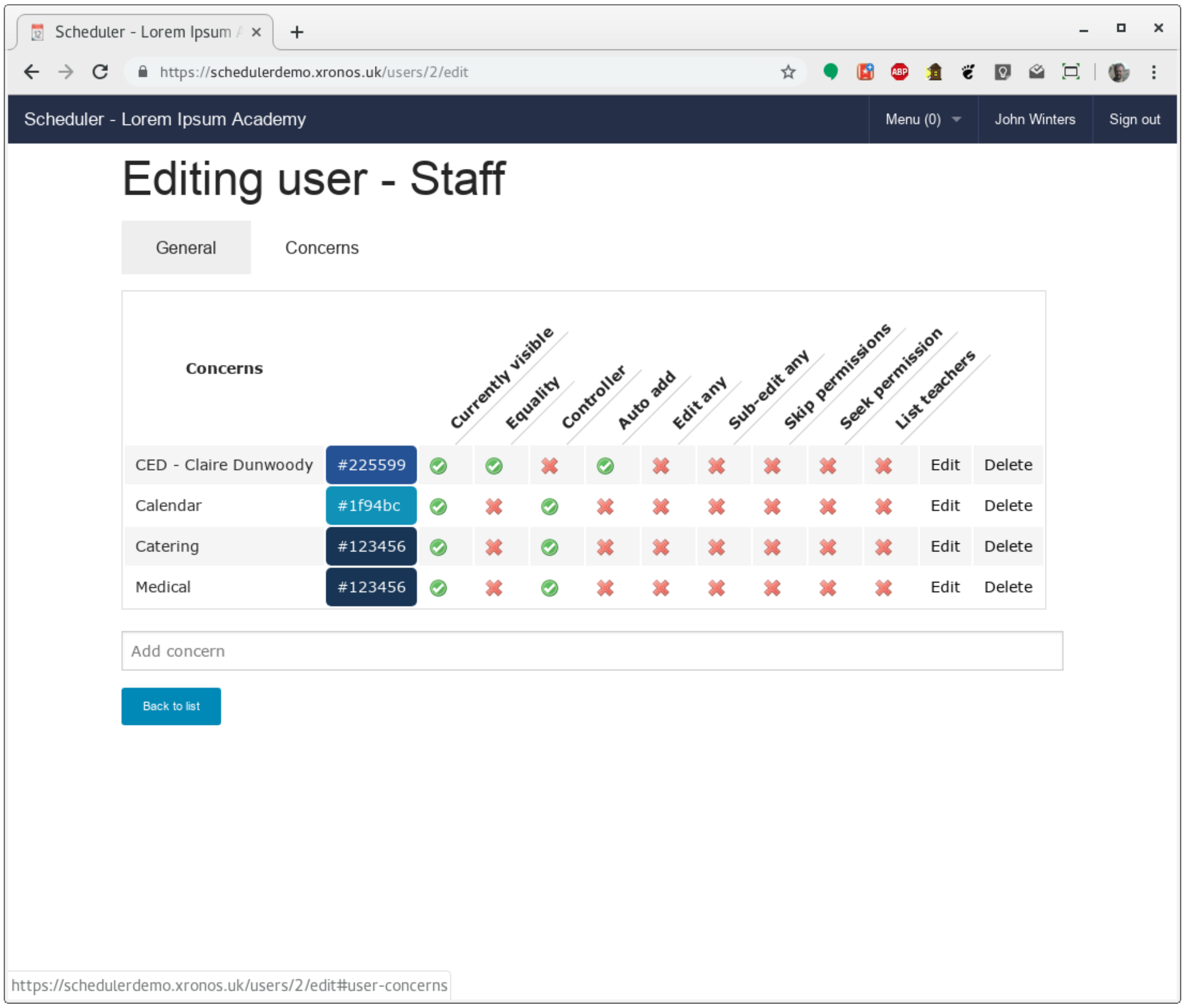
Creating events
☑ Warn resources ☑ Show pre-requisites

Clash notifications
☐ Weekly ☐ Daily ☐ Immediate

Invigilation notifications
☑ Weekly ☑ Daily

[Back to list] [Update]

Notice that there are two tabs here, labelled “General” and “Concerns”. Click on the “Concerns” tab to see the existing links between CED and resources within the system.



Scheduler - Lorem Ipsum Academy

Editing user - Staff

General | Concerns

Concerns		Currently visible	Equality	Controller	Auto add	Edit any	Sub-edit any	Skip permissions	Seek permission	List teachers		
CED - Claire Dunwoody	#225599	✓	✓	✗	✓	✗	✗	✗	✗	✗	Edit	Delete
Calendar	#1f94bc	✓	✗	✓	✗	✗	✗	✗	✗	✗	Edit	Delete
Catering	#123456	✓	✗	✓	✗	✗	✗	✗	✗	✗	Edit	Delete
Medical	#123456	✓	✗	✓	✗	✗	✗	✗	✗	✗	Edit	Delete

Add concern

Back to list

https://schedulerdemo.xronos.uk/users/2/edit#user-concerns

CED is currently linked to four resources within the system.

- Herself
- The public calendar
- Catering
- Medical

Those last two are Services within the demonstration system.

All of them are currently visible - that is, events involving them will appear on her default Scheduler display.

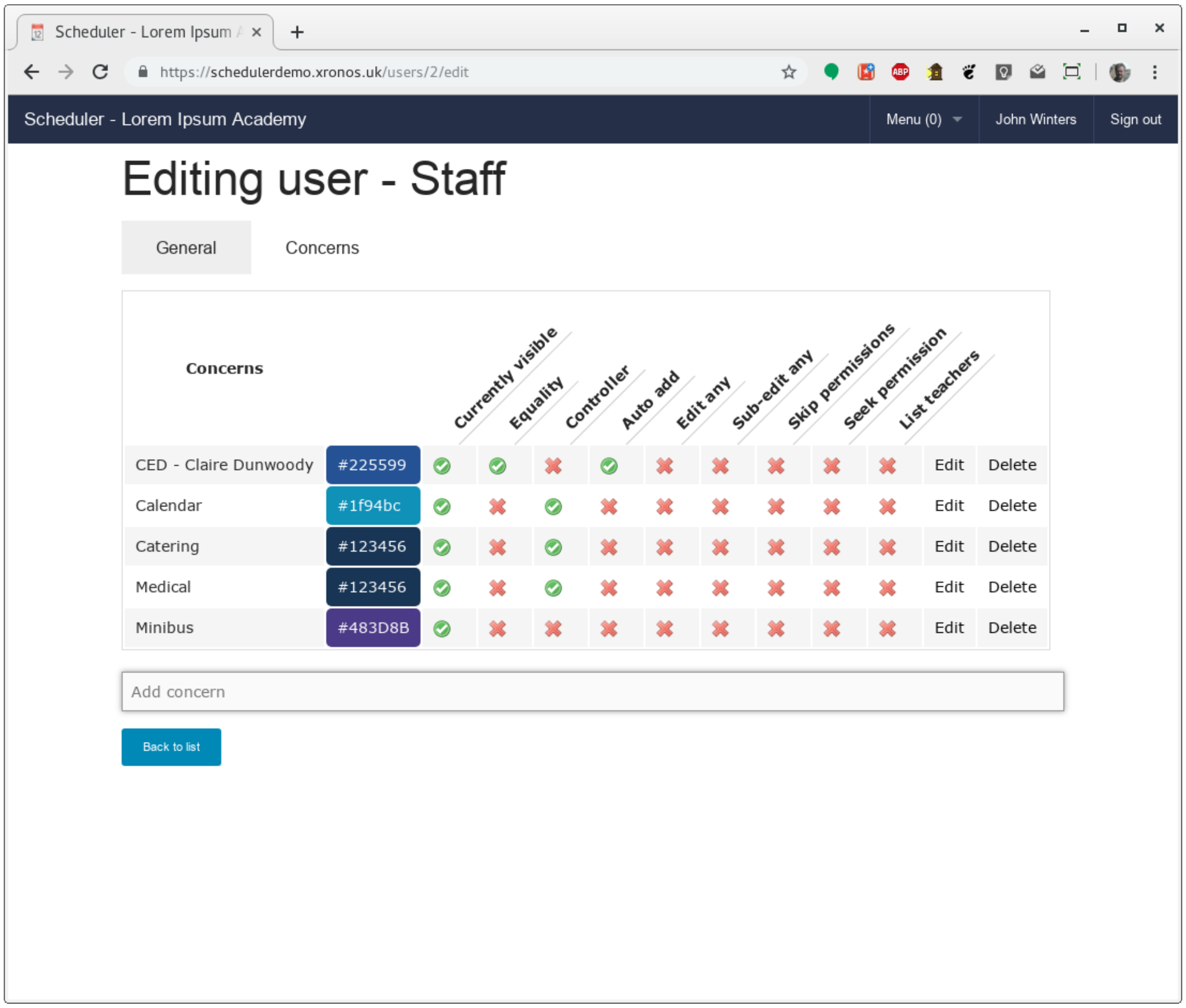
For three of them though she is also listed as being a “Controller”. This has two effects:

- The corresponding resource is controlled, meaning that it can’t just be added to an event by other users - it goes through the permissions process
- CED can grant permission for such requests

Let’s say we want to make CED a controller of minibuses too. First she needs a Concern linking her user record to the Minibus resource group. She could add this herself on the main screen, but the system administrator can add it here too.

Just type “Minibus” in the “Add concern” box below the Concerns listing - the usual predictive text will appear. Select the required item and press Enter.

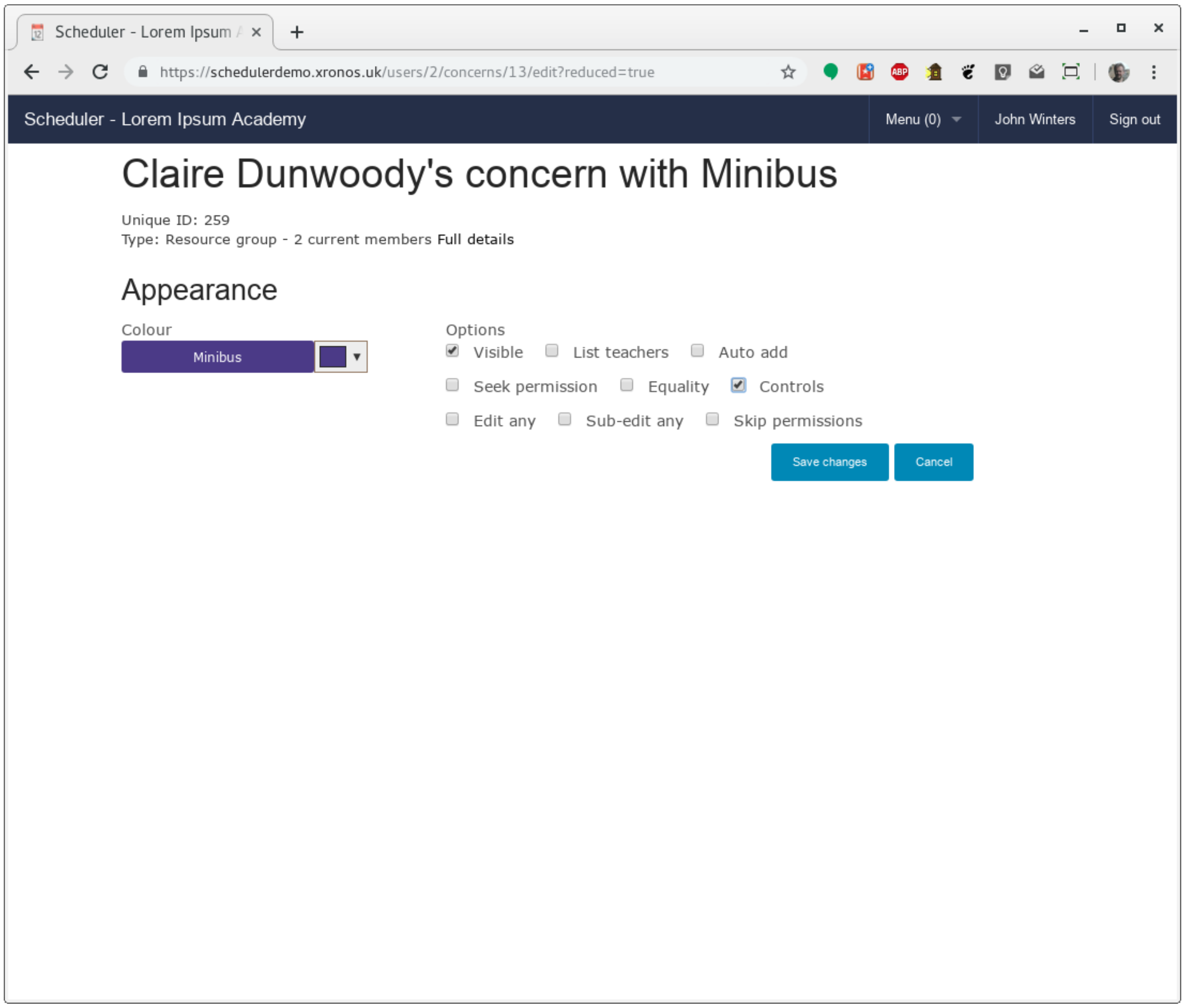
CED then has an extra Concern, like this:


Scheduler - Lorem Ipsum A
https://schedulerdemo.xronos.uk/users/2/edit
Scheduler - Lorem Ipsum Academy
Menu (0)
John Winters
Sign out
General
Concerns
Add concern
Back to list

Editing user - Staff

Concerns		Currently visible	Equality	Controller	Auto add	Edit any	Sub-edit any	Skip permissions	Seek permission	List teachers		
CED - Claire Dunwoody	#225599	✓	✓	✗	✓	✗	✗	✗	✗	✗	Edit	Delete
Calendar	#1f94bc	✓	✗	✓	✗	✗	✗	✗	✗	✗	Edit	Delete
Catering	#123456	✓	✗	✓	✗	✗	✗	✗	✗	✗	Edit	Delete
Medical	#123456	✓	✗	✓	✗	✗	✗	✗	✗	✗	Edit	Delete
Minibus	#483D8B	✓	✗	✗	✗	✗	✗	✗	✗	✗	Edit	Delete

Click on the “Edit” link for the Minibus concern, and tick the “Controls” check box.



Scheduler - Lorem Ipsum Academy

Claire Dunwoody's concern with Minibus

Unique ID: 259
Type: Resource group - 2 current members Full details

Appearance

Colour: Minibus

Options: [✓ Visible] List teachers Auto add Seek permission Equality [✓ Controls] Edit any Sub-edit any Skip permissions

Then click "Save changes" to put your change into effect.

CED is now a controller for the "Minibus" resource, and requests for a minibus will come to her.

2.3 Default resources

A number of default Properties are created when a new Scheduler system is installed. These are expected to exist for the proper running of the system.

Several of these almost certainly need to be given controllers before your system goes live. Specifically:

- Calendar
- Suspension
- Gap

The "Calendar" property is for events which you want to appear in your public calendar - events which are to be visible to users who have not logged in to your system. This should be under the control of the person who decides what goes in your public calendar.

Note: Although there is just one “Calendar” in a new system, you can create as many public calendars as you like, and you can rename the original one. Just create a Property and tick the “Public” check-box for it.

If you have more than one public calendar in your system, Scheduler lets guests choose which ones they will see.

The Suspension and Gap properties are used to massage the timetable as it is imported from your MIS. For full details see the documentation on *Gaps and Suspensions*

Again, you don’t want ordinary users to be able to allocate these properties to their events, so give them a controller.

CHAPTER

THREE

LOCATIONS (AND ALIASES)

Scheduler uses the term “Location” to refer to any location where you might want to schedule events - these might be classrooms, meeting rooms, halls, fields, or anything you choose.

In dealing with locations, it is important to understand also the concept of a “Location Alias” - why they exist and what they do for you.

3.1 Locations

The location records within Scheduler are fairly simple and straightforward to understand. Each one has a name, and refers to a real identifiable place within your organisation. Typically the name will be fairly short and could be a classroom identifier (e.g. SB112) or other short name.

The internal event processing of Scheduler uses location records *only* - it has no interest at all in Location Aliases. Location records are attached to events in just the same way as records relating to people, services etc.

Each real place within your organisation should have one (and only one) Location record within Scheduler.

A location with no aliases (see below) will always be referred to by the name in its Location record. Adding aliases can affect the way the name is displayed (and searched for), but never has any effect on the Location record itself, nor how it is attached to events.

One place - one Location record.

3.2 Location Aliases

Location Aliases exist within Scheduler to cope with several related issues.

- A location might be stored in the MIS with a less than friendly name.
- A location might be stored in the MIS more than once under different names. (It happens!)
- A location might be referred to by more than one name within the school. E.g. as SB112 when used as a classroom, but as “Governors’ Meeting Room” when used for meetings.

Each Location Alias record contains a name, and is linked to a Location record. The name in the Location Alias record need not be the same as the name in the Location record - and indeed it often isn’t. One Location record can have as many Location Alias records as is needed.

To cope with the first issue, data import from the school’s MIS is always done by way of a Location Alias record. Each reference in the MIS to a location causes Scheduler’s importer to find a corresponding Location Alias, and thus the correct Location record. The name stored in the Location record can be different from the one used by the MIS and the lookup will still work.

Note: In reality, after the initial record creation, the Location Alias record will be found not by name but by reference number. It is however useful to retain the MIS's idea of the location's name in the Location Alias record for documentation purposes.

Very similar processing takes care of the second issue. If a single location exists more than once in the MIS, then Scheduler copes by having the appropriate number of Location Alias records, each matching one instance in the MIS, and all linked to the same Location record. See the section below on *initial import* for details on how to set this up.

Each time the MIS refers to any of its duplicate entries, Scheduler will understand that they all mean the same place.

For the case where users refer to the same location by two or more different names, some additional facilities of the Location Alias record come into play.

Imagine a school has a library, which also has a classroom number - say, BR218. People at the school might refer to it by either name, but it's the same place.

For this instance, the Location record should be set up with a name of "BR218", and then a Location Alias record should be created with the name "Library", and its "Display" flag ticked. This will cause the system as a whole, when wanting to refer to the location by a long name, as "BR218 / Library". This name is referred to as its "Display name" and is constructed by taking the name from the Location record, and following it with the names of each of the Location Alias records which have the "Display" flag ticked.

Any searches will be fulfilled using this same display name, so users can find the location in Scheduler using either name.

Scheduler also has the concept of a long or "Friendly" name for a location. Using the earlier example of SB112 being the Governers' Meeting Room, it might also be referred to by the abbreviation "GMR". It could be set up in Scheduler as follows:

- Location record containing the name "SB112"
- Location alias containing the name "GMR", with "Display" ticked.
- Location alias containing the name "Governors' Meeting Room", with both "Display" and "Friendly" ticked.

This will result in a display name of "SB112 / GMR / Governors' Meeting Room", making it clear that they are all the same place, and allowing searches for any of these strings to find it.

Reports intended for the general public to read (e.g. the public view of a public calendar) will use the friendly version of a location's name. In this case, the public will see "Governors' Meeting Room" rather than an internal acronym like GMR, which might not be understood.

Code which needs a short name for a location (e.g. code to print a timetable) will always use just the name from the Location record and ignore any aliases. This is why it makes sense to ensure that each Location record contains a short, clear name for its location.

3.3 Initial MIS import

When you first do a data import from your MIS, it will cause the creation of a lot of pairs of Location and Location Alias records. It's best to do this before you start manually creating Location records. After the initial import, you can add in any locations which the MIS doesn't know about.

Note: The MIS importer creates Location and Location Alias records as pairs - one for each location in the MIS. The Location Alias record in this pair exists solely so that it can find the same record for subsequent imports, even if you've renamed the Location, or even deleted it and linked the Location Alias to a different Location.

When you create your own Location records you don't need to create any corresponding Location Alias records, unless you want the Location to have multiple names.

If there is a location which appears twice in the MIS but is really just one place then you can merge the records after the initial import.

Take the example of BR218 being the library as well, and let's assume it appears as two separate locations in the MIS. After the initial import you will have:

- Location record, named BR218
- Location Alias, also named BR218, linked to the Location BR218
- Location record, named Library
- Location Alias, also named Library, linked to the Location Library

We want to keep both those Location Alias records, because they will be used in subsequent data imports, but we want only one Location record. We also want all events which reference either of the separate Location records to reference instead the one merged record.

Happily, there is a helper method which will do all this for you, although it does need to be invoked from the command line.

Decide which one makes the best short name - in this case BR218. You then absorb the other location record into it.

```
$ . ~/etc/whichsystem
$ cd $SCHEDULER_DIR
$ rails c
Loading production environment (Rails 4.2.10)
2.3.6 :001 > l = Location.find_by(name: "BR218")
...
2.3.6 :002 > l2 = Location.find_by(name: "Library")
...
2.3.6 :003 > l.absorb(l2)
...
Absorbed 121 commitments and 1 alias.
Deleting Library
=> nil
2.3.6 :003 > exit
$
```

Note that the location which you're keeping absorbs the one which is going away. After this process, your database records will be as follows:

- Location record, named BR218
- Location Alias, also named BR218, linked to the Location BR218
- Location Alias, named Library, linked to the Location BR218

So any future imports will direct all events to the one single location.

CHAPTER FOUR

GAPS AND SUSPENSIONS

4.1 Terminology

The average MIS will store your school's timetable and tell you what would be happening in a perfect week within the school.

Scheduler aims to go one better and tell you what *is* happening this week (or any particular week).

Sometimes some timetabled lessons do not happen - either for a particular year group or for the entire school. This might be for a wide range of reasons, e.g.

- The whole school is doing a sponsored walk
- Year 11 are on study leave
- Year 6 are going to the Science Museum

For the first of these - the whole school sponsored walk - the entire timetable is simply not going to happen. This is what Scheduler calls a **Gap**. For that particular day, all lessons should simply disappear.

The other two are slightly more subtle. Lessons will still be happening, although some of them won't, and it can be useful to know what lessons *would* have happened if there hadn't been this special event. For the case of study leave, it might well be that the staff just freed up are then needed for supervised study periods or exam invigilation. For the year 6 trip, staff who would otherwise have been teaching year 6 might be used to cover the lessons of staff accompanying the trip.

For this reason, Scheduler can keep the affected lessons visible in the system, but mark them as not actually happening. In Scheduler's terms, this is a **Suspension**.

By setting up suitable events within Scheduler, the system administrator can cause these modifications to the timetable. The modifications are effected each time the MIS importer runs - usually each night.

Note: Gaps and Suspensions do not affect events entered by users into the system - just the academic and activity timetables brought in through an automated import.

4.2 Controlling

Obviously it isn't a good thing for every user of the system to be able to create Gaps and Suspensions on a whim. Chaos would ensue.

There are two special Properties within Scheduler called "Gap" and "Suspension" which are used to implement the functionality described here.

You should protect these two Properties by setting up users to control them as detailed in *Controlling resources*

4.3 Creating

When setting up a **Gap** or a **Suspension**, the first thing is to decide whether it affects the whole school or just one year group.

If you apply one of the two properties - Gap or Suspension - to an event with no students attached then it will affect the entire school. This would be appropriate for something like the Whole School Sponsored Walk mentioned above.

If there are any students also attached to the event, then the importer will make a list of their year groups, and apply the gap or suspension to all those year groups.

To suspend lessons for year 11, it is therefore sufficient to attach just one year 11 student to the suspension event. It makes things a bit clearer though to attach the group "Year 11 students".

What you need therefore to create a **Gap** or **Suspension** is an event with the appropriate duration, with either the Gap or the Suspension property attached to it. To restrict it to just one year group, attach some students as well.

Note: You can either attach the Gap/Suspension property to an existing event, or you can set one up specially. Which you do depends on whether there is a fully suitable event already in the system.

For the Whole School Sponsored Walk mentioned earlier, there will probably already be an event in the public calendar saying that it's happening. One can therefore just attach the "Gap" property to this event.

On the other hand, for Year 11 study leave it might be the case that students can still come in to school, but have to register. Registration lessons are thus still needed, but ordinary lessons are not happening. The event "Year 11 Study Leave" in the school's public calendar just has a start date and end date so it's not subtle enough.

Instead one could create a hidden repeating event, running from 09:20 to 16:00 on each date in that range, with the "Suspension" property and "Year 11 pupils" attached to it. That way all the lessons from 09:20 to 16:00 for year 11 pupils would be suspended, but registration (which runs from 09:00 to 09:20) would not. The event would be placed in the Hidden event category, so no-one sees it apart from its creator.

CHAPTER

FIVE

AD HOC LESSONS

5.1 Purpose

Scheduler uses the term “Ad Hoc Lessons” to refer to lessons where pupils are taken out of their normal timetabled lesson for some kind of extra tuition. Typically this will be an instrumental music lesson, learning support or something similar.

It is generally desirable to rotate these kinds of lessons so pupils don’t miss the same subject more often than is necessary. If, for instance, an instrumental music lesson makes a pupil miss a maths lesson one week, then the same pupil should not miss another maths lesson the following week.

The person responsible for scheduling these Ad Hoc lessons therefore needs access to all the pupils’ individual timetables, plus a record of what lessons they have already missed. Scheduler aims to assist with this by providing all the relevant information in one place.

Once a suitable schedule for the term (or half term, or whatever period you choose) has been devised it can be published automatically to both pupils and relevant teachers.

5.2 Configuration

It is up to the system administrator to define categories of Ad Hoc Lessons and appoint controllers for each category. Once this is done, the rest of the scheduling work can be done by those controllers.

Each category needs:

- A name - for instance, “Music lessons”.
- An event source - to identify all its lessons.
- An event category in which to place its events - typically “Ad Hoc Lesson”.
- A default day shape to define when lessons might occur.
- An identifying Property - to allow users to select such lessons to view.

The system administrator also needs to define (because they are global items within the system):

- The day shape to be used by the Ad Hoc category.
- Any Subject records - e.g. Violin, Viola, Mandolin, etc.

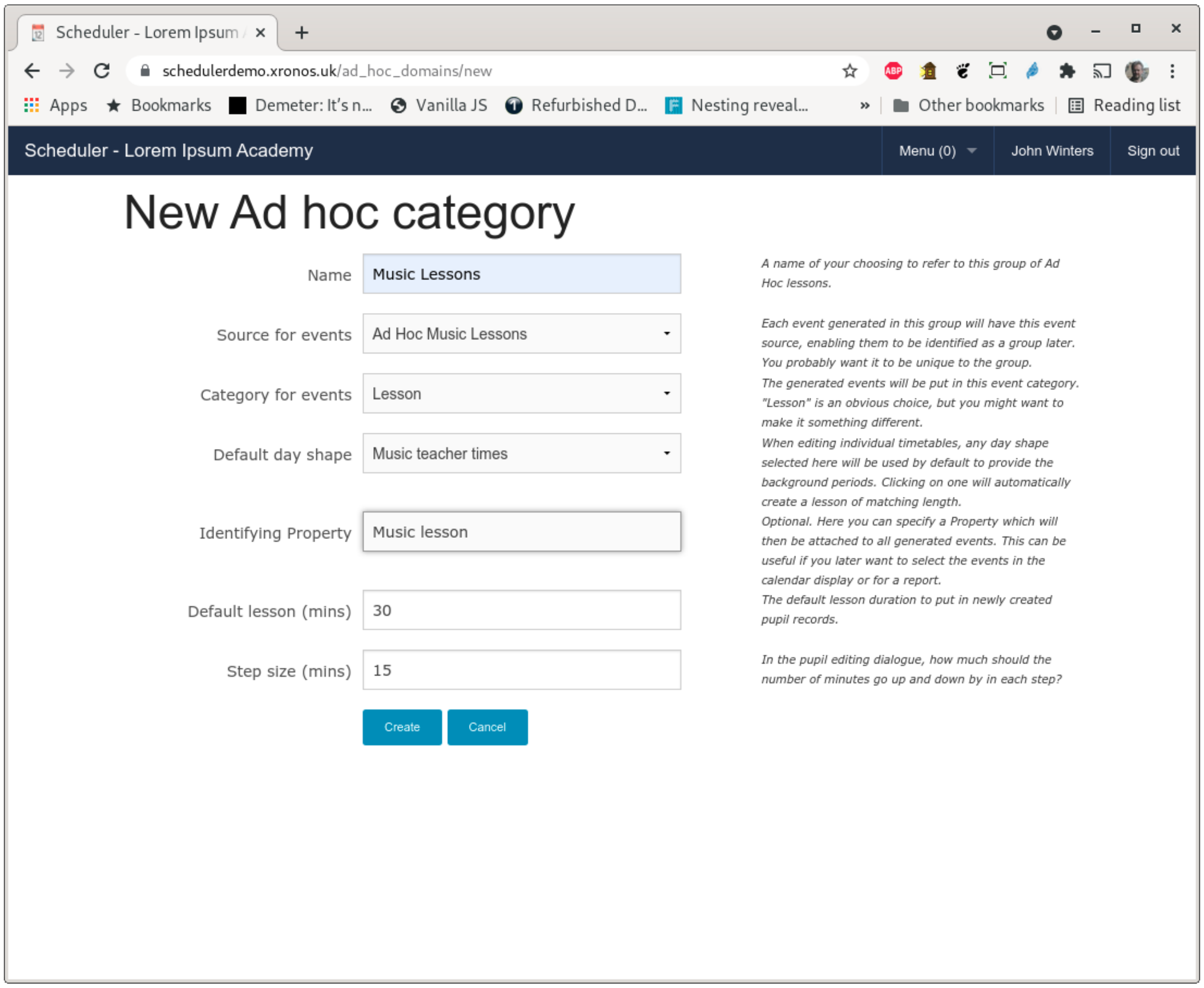
Start by creating a Day Shape (Menu => Admin => Day shape) if required. Typically each teacher of this sort of lesson will have a different availability (e.g. Monday all day plus Tuesday mornings) but if you have an overall shape to the day (e.g. Mondays are from 09:00 to 12:30 and afternoons are from 14:00 to 16:00) then you can create a corresponding Day Shape which will then assist the controller of the Ad Hoc category in setting up individual availability records for each teacher.

Create an Event Source using Menu => Admin => Models => Events => Sources

Create an Event Category using Menu => Admin => Models => Events => Categories. It is important that Ad Hoc Lessons have their own category because they want slightly different flags from ordinary lessons. When the nightly job runs to create lists of pupils who will be missing lessons, we want it to flag pupils who are missing ordinary lessons for Ad Hoc lessons, but not generally the other way around. The whole point of Ad Hoc lessons is that they override the normal lessons.

Create a new Property using Menu => Admin => Models => Properties

You can then create your new Ad Hoc category using Menu => Admin => Ad Hoc t/ts and then click on "New category" which will give you a screen like this:

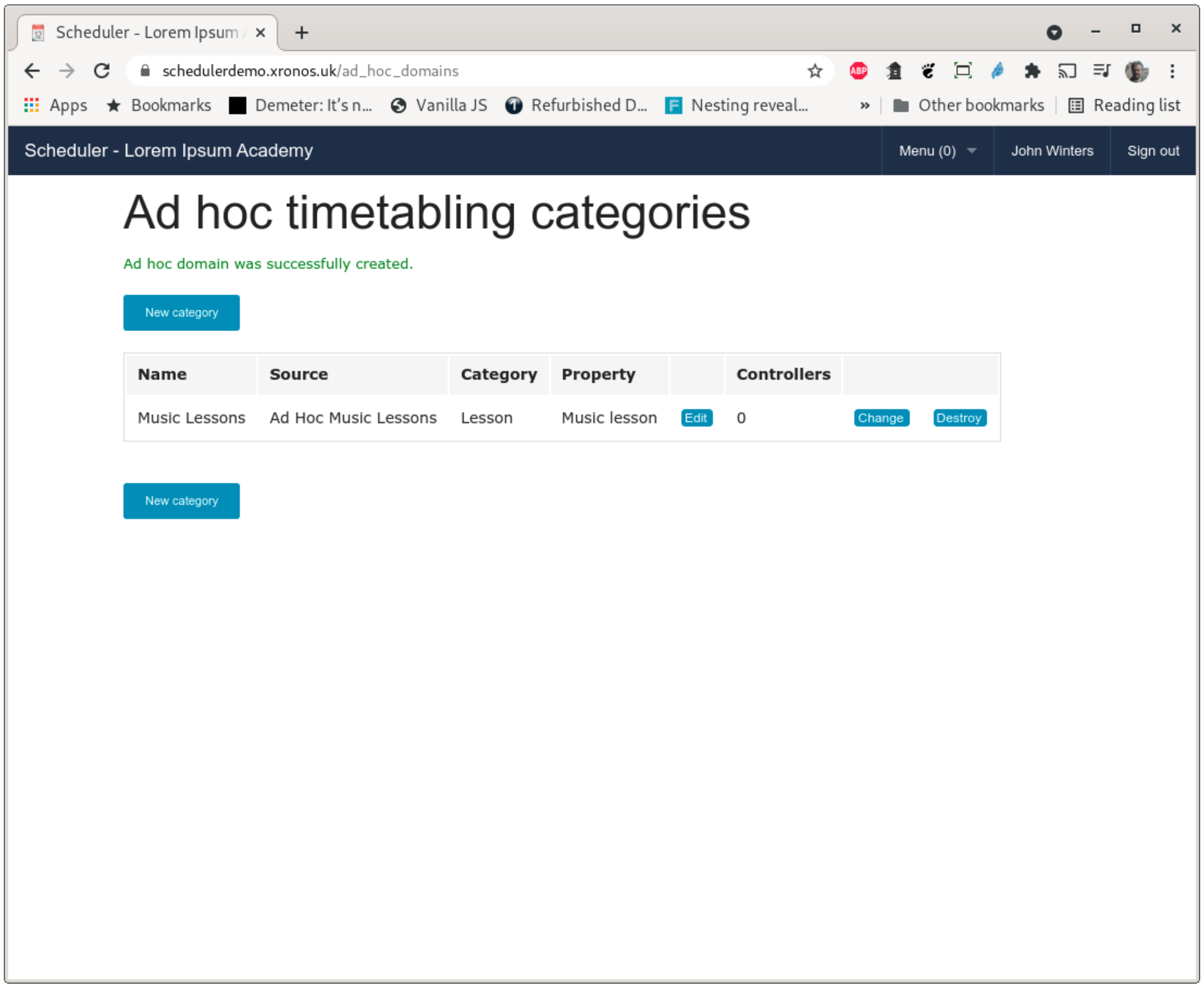

Scheduler - Lorem Ipsum
schedulerdemo.xronos.uk/ad_hoc_domains/new
Scheduler - Lorem Ipsum Academy
Menu (0)
John Winters
Sign out

New Ad hoc category

Name	Music Lessons	A name of your choosing to refer to this group of Ad Hoc lessons.
Source for events	Ad Hoc Music Lessons	Each event generated in this group will have this event source, enabling them to be identified as a group later. You probably want it to be unique to the group.
Category for events	Lesson	The generated events will be put in this event category. "Lesson" is an obvious choice, but you might want to make it something different.
Default day shape	Music teacher times	When editing individual timetables, any day shape selected here will be used by default to provide the background periods. Clicking on one will automatically create a lesson of matching length.
Identifying Property	Music lesson	Optional. Here you can specify a Property which will then be attached to all generated events. This can be useful if you later want to select the events in the calendar display or for a report.
Default lesson (mins)	30	The default lesson duration to put in newly created pupil records.
Step size (mins)	15	In the pupil editing dialogue, how much should the number of minutes go up and down by in each step?

Note that this form has been filled in using the records referenced previously. You can also specify the normal length of a lesson, plus how much it might be adjusted by. For this example, the normal length of a lesson is 30 minutes, but it might be desired to create one of 45 or even 60 minutes, so the step size has been set to 15 minutes.

Hit the "Create" button and you will be returned to the listing of Ad Hoc categories showing your newly created one.



Ad hoc timetabling categories

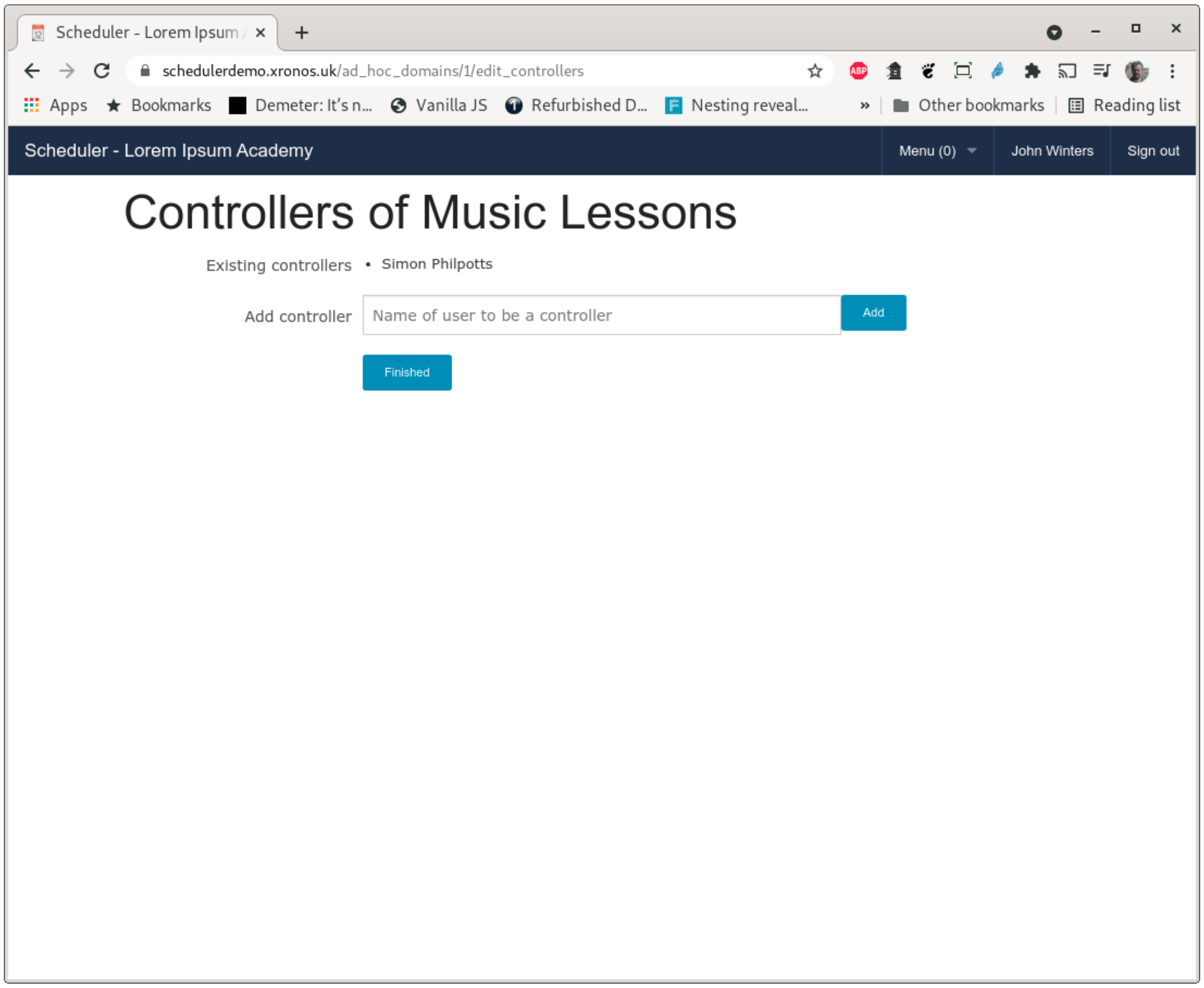
Ad hoc domain was successfully created.

New category

Name	Source	Category	Property		Controllers		
Music Lessons	Ad Hoc Music Lessons	Lesson	Music lesson	Edit	0	Change	Destroy

New category

Appoint at least one Controller for the category by clicking the “Change” button.



Controllers of Music Lessons

Existing controllers • Simon Philpotts

Add controller: Name of user to be a controller [Add]

[Finished]

In this case, the ubiquitous Simon Philpotts is taking on the rôle again.

You have now set up the Ad Hoc category and the rest of the work can be done by the Controller(s). Documentation for them will be found in the Scheduler Advanced User Guide

CHAPTER

SIX

END OF YEAR

6.1 MIS feed

It is likely that your Scheduler installation takes a regular feed from your school's main MIS (SchoolBase, iSAMS or WCBS/Pass) in order to keep its idea of the timetable and teaching groups up to date.

Soon after the end of the academic year, the information in the MIS is liable to enter a state of flux. Firstly there will be some kind of end-of-year processing within the MIS, ending one academic year and starting another. Then the person responsible for your school's timetable will start entering the new one for the coming academic year, and all the old teaching sets will be scrubbed (or promoted by a year) and new ones created and populated.

It is important to make sure that your MIS and Scheduler have a common idea of which academic year is current whenever information is fetched across. For this reason, it is recommended that the following steps are carried out:

- Stop the cron job fetching data from the MIS to Scheduler. This is best done immediately after your term has ended.
- Perform end-of-year processing in your MIS.
- Perform end-of-year processing in Scheduler.
- Wait until the timetable is reasonably stable and sets have been set up for the new year.
- Re-enable the cron job for fetching data.

One would typically expect the cron job to be re-enabled around the middle of August, assuming your term starts in early September. Don't start it before the start date of the era representing your new academic year. See below for more about eras.

6.2 Eras

Eras exist in Scheduler primarily for the purpose of keeping track of groups. Typically you will set up one era per academic year - e.g. "Academic Year 2017/18" - although if you have an institution where the teaching groups change dramatically each term or semester then you could use one era per term/semester.

Each group within Scheduler belongs to an era. This is because it is necessary to distinguish between groups of the same name from different eras. Typically you will have a top set for maths in year 11 each academic year, and they will all have the same name - perhaps 11MAT1. Although two of these from consecutive academic years have the same name, they are quite different groups - different pupils and probably a different teacher. We don't want them to get muddled up and so all teaching groups belong to the current era, and will end when the era ends.

Some other groups have continuity through the academic years. For instance you might have a group called "Housemasters", or "1st Orchestra". These will vary a bit from year to year, but a large chunk of the membership remain will remain the same. Groups like this belong to the Perpetual era (typically called "Perpetual Era") and carry on through the years. Their membership can be adjusted as required and the system will keep a permanent record of who was in the group when. It is thus possible to answer questions like "Who was in the 1st Orchestra at the beginning of July last year?"

6.3 Preparation

If you are in the position of wanting to perform end-of-year processing in Scheduler then it will probably already have the following four eras set up:

- Previous era
- Current era
- Next era
- Perpetual era

Take a look at them (Menu => Settings to see which they are, and Menu => Models => Eras to see their dates).

Typically your "Current era" will be set up to end at about the middle of August (again, assuming an academic era starting in September) and the "Next era" will be set up to start on the following day. You can if you wish adjust these but it's probably not a good idea to make the eras overlap.

Looking ahead a few steps - don't restart the data feed from MIS => Scheduler until after the start of what is currently your "Next era". If you want to start it earlier than mid-August, then now would be a good time to adjust the end date of the current era and the start date of the next era.

6.4 Back ups

Take a fresh full backup of your entire Scheduler database. Put it in a safe place.

Warning: Seriously - do make sure you have an up-to-date backup before you start. Much pain can be avoided in the event of finger trouble or whatever if only there is a good backup.

6.5 Processing

The actual end-of-year processing is currently performed from the command line. Once you're happy with the dates of your "Current" and "Next" eras, and you have a good backup, log in as your scheduler user and proceed as follows:

```
$ . ~/etc/whichsystem
$ cd $SCHEDULER_DIR
$ rails c
Loading production environment (Rails 4.2.10)
2.3.6 :001 > Setting.first.end_of_era
Rolled over.
=> nil
2.3.6 :002 > exit
$
```

Since this goes through closing out all your existing groups attached to the current era, it may take 20-30 seconds to run.

6.6 Tidy up

If you now go back and look at your system settings again, you should find that what was your “Current era” is now your “Previous era”, and what was your “Next era” is now your “Current era”.

This is quite a good time to create a new era for the following academic year and then configure it in the system settings as the new “Next era”. You can always tweak its dates nearer the time of the next roll-over.

You probably don’t want to re-enable the MIS import immediately (and in any case, don’t do it before the start date of what is now your “Current era”). Wait until the information in the MIS has stabilized a bit. You don’t need all the sets to be there, but you really don’t want to be downloading incorrect information which will just need to be changed later.

The first run of the importer after the summer hiatus will take much longer than normal. A normal run *checks* every timetable entry and group membership. This run will need to *create* them all, and creating a record in a database is a much slower job than reading one. Pick a time when the system is quiet, and allow several hours for the run to complete.

CHAPTER

SEVEN

UPGRADING THE APPLICATION

. . . or, upgrading the software of Scheduler itself.

The home of the Scheduler software is GitHub

If you already have a working Scheduler installation then this information should be embedded within it but for reference, the URL for fetching the software is https://github.com/XronosSchedulingLtd/scheduler.git

At any given moment, the "master" branch on GitHub will point to the latest released version, whilst in addition there are tags for each release, of the form "v1.2.0".

You might choose to track the master branch (this is the default option) or you can lock your installation to a chosen version. The latter is useful if you need to do a progressive upgrade from a very old version - see below.

7.1 Cron jobs

Before you start to do an upgrade, think about when your various cron jobs are configured to run. They happen in the background and some of them can take quite a few hours to run - those which check all the events for student clashes and flag them.

If you start doing an upgrade whilst a cron job is running, odd things could happen.

7.2 Release notes

Before any upgrade, read the release notes. These are in a file called RELEASE_NOTES in the root directory of the application, and you can read it directly on GitHub.

There may be specific instructions there relating to a particular release. Release v0.28.3 is of particular note because it involves switching to a later version of Ruby, and thus needs extra steps.

7.3 Backup

Before you start, always take a fresh backup of your database. You know it makes sense.

7.4 Check current version

Before you start you need to know two things:

- What version of software do you currently have?
- What version is your system set up to request?

The first of these is easy - unless you have a very old version of the software, the version number is displayed in the bottom right-hand corner of the main screen of the application.

Note: If you have admin privileges within the application you can also get additional information by hovering your mouse pointer over this version number. The hover text will tell you the both the version of Ruby and the version of Rails in use.

If your version is too old to display the version number like this then you will find it in a file called "VERSION" in the root directory of the application.

Then you want to know what version of software your system is currently configured to expect. This will affect how it behaves when you connect to GitHub for an update.

The 'git status' command will give you this information. As the scheduler user, change into the SCHEDULER_DIR and type 'git status' and you should see something like this:

```
$ . ~/etc/whichsystem
$ cd $SCHEDULER_DIR
$ git status
On branch master
Your branch is up-to-date with 'origin/master'
nothing to commit, working directory clean
$
```

If you have changed any local configuration files that last line won't appear and instead you will see a list of the changed files.

Note: The "Your branch is up-to-date. . . " line is potentially confusing. It doesn't mean that your branch is necessarily up-to-date with the 'master' branch held on GitHub. 'master' and 'origin/master' are both branches on your local machine. What it means is, the last time your system checked with GitHub it was up to date.

The interesting line of output however is the first one - "On branch master". That tells us that our system is set up to follow the latest release from GitHub, even if we haven't got it yet. If instead you get something like this:

```
$ git status
HEAD detached at v1.2.3
...
```

Then your system is currently locked to version 1.2.3 of the software. We'll see why you might want to have it like this in upcoming sections of this page.

7.5 Basic steps

Assuming your system is on branch 'master' and you simply need to upgrade by one version, the steps are as follows:

- Stop the application
- Install the new version of software from GitHub
- Install any required support packages
- Apply any required database structure changes
- Pre-compile the CSS and JavaScript assets
- Re-start the application

Note that some care is needed if you have let your installation get out of date. It is not always possible to go straight from a very old version to the current version. See below for instructions on how to update a very old version.

For this simple case, your upgrade session should look like this:

```
$ . ~/etc/whichsystem
$ cd $SCHEDULER_DIR
$ bin/delayed_job stop
$ sudo systemctl stop puma
[sudo] password for scheduler:
$ git pull
$ bundle install
$ rake db:migrate
$ rake assets:precompile
$ sudo systemctl start puma
$ bin/delayed_job start
```

Note that, unless you've taken a very long time over this, you won't be prompted for your password a second time.

Note: The "git pull" command both fetches the new version from GitHub and places it into your application's root directory. That's fine for this simple upgrade case.

7.6 Upgrading in steps

If your system is more than one release behind GitHub then the general rule is to upgrade one release at a time. If you have only two or three to do then this isn't too hard. If you have more, then see the next section.

Note: Upgrading by several versions all in one go may be fine, but there are circumstances in which it will not work.

Imagine you're on version 1 of the software.

The migration to version 2 causes an update to all the user records.

The migration to version 3 introduces a new field to the user record *and* adds an application software constraint that this field must always be populated. It then immediately populates the field.

If you try to go straight from version 1 to version 3, then the new (version 3) software will be installed before any of the database updates are done. Then the database migrations are executed sequentially and the one for version 2 will fail because the constraint requiring the new field to be populated is already in place but the field does not exist.

If however you upgrade from version 1 to version 2, then from version 2 to version 3, all will be well.

Let's assume you are currently running v1.2.8 of the software and the current version is v1.2.11. We'll assume further that your installation is on the 'master' branch - you just haven't got around to updating for a while.

Warning: At this point, your system's idea of what the 'master' branch is differs from GitHub's. We need to bring them gently back into line.

Start by shutting down nginx (as above) and then lock your system to v1.2.8 of the software. Once you've done that you can bring down all the new versions from GitHub:

```
$ . ~/etc/whichsystem
$ cd $SCHEDULER_DIR
$ bin/delayed_job stop
$ sudo systemctl stop puma
[sudo] password for scheduler:
$ git checkout v1.2.8
$ git fetch
```

Note the use of "git fetch" instead of "git pull". "git pull" says, "Grab the latest of everything and put my chosen version in my application directory." If you do that whilst your system is on the 'master' branch then you'll jump straight to the latest version - not what is wanted in this case.

The "git fetch" command on the other hand says, "Fetch down anything new which you can find on GitHub, but don't do anything with it for now - just fetch it".

You're now in a position to apply the updates one by one. Still working on our imagined task of going from v1.2.8 to v1.2.11 you would do the following:

```
$ git checkout v1.2.9
$ bundle install
$ rake db:migrate
$ git checkout v1.2.10
$ bundle install
$ rake db:migrate
$ git checkout v1.2.11
$ bundle install
$ rake db:migrate
$ rake assets:precompile
```

And then assuming that v1.2.11 is indeed the latest version we can put the system back into its original state with:

```
$ git checkout master
$ sudo systemctl start puma
$ bin/delayed_job start
```

Note: When you check out an explicit version like v1.2.9 you get what looks like a dire warning from Git about your system being in a "Detached HEAD" state. This is not a problem, because you're not doing development work on the system - just installing specific versions of the code.

If you have set yourself up with a test system then you can short-circuit this process by trying the update on your test system first. If all the database migrations run without error then it will be fine to go straight from your start version to your target version (but see note about v0.28.3 below).

CHAPTER

EIGHT

UPGRADING RUBY

How to upgrade the Ruby interpreter used by Scheduler.

Warning: All the usual caveats about having a good backup of your system and database apply. Just do it.

Sometimes a new version of Scheduler will require a new version of Ruby as well. Ruby is an evolving language, and much work goes in to adding new features, improving the speed and fixing security issues.

The Scheduler software aims to make use of a reasonably up-to-date version of Ruby, whilst avoiding the bleeding edge.

When a new version of Scheduler becomes available, the release notes will indicate whether it also requires a new version of Ruby. This is an unusual occurence, so the steps described here are not generally needed.

Note: Stick to the version of Ruby recommended by Scheduler. Much testing is done to ensure compatibility, and whilst Scheduler itself is not terribly version sensitive, some of the supporting packages can be.

The current recommended version of Ruby is 2.5.5, and the previous recommended one was 2.3.6.

These instructions are written on the assumption that you're upgrading Ruby from version 2.3.6 to version 2.5.5, but you should adjust them appropriately if you're installing a later version of Scheduler which wants a later version of Ruby.

8.1 Preparatory steps

One convenience of the upgrade process is that you can do some of it in advance, without disturbing your running Scheduler system.

We will assume here that you're about to install Scheduler version 1.18.0, which is the first version which requires Ruby 2.5.5. Again, adjust for later versions.

The Ruby versions used by Scheduler are managed by the Ruby Version Manager (RVM) which allows you to have several versions installed at the same time.

As a first step, it's a good idea to make sure you have the latest stable version of RVM.

```
$ rvm get stable
```

Note that this is done as your Scheduler user, not as root. All the Ruby stuff is kept in the user's area.

Note: If you have currently a very old version of RVM, it’s possible that the above command will fail with a message saying that it is not possible to verify the signature on the new package.

If that happens, you need to get the latest signing keys for RVM, using the following command.

```
gpg --keyserver hkp://pool.sks-keyservers.net --recv-keys␣
↪409B6B1796C275462A1703113804BB82D39DC0E3 7D2BAF1CF37B13E2069D6956105BD0E739499BDB
```

Once that has completed, repeat the `rvm get stable` command.

With the latest version of RVM installed, installing Ruby is just a question of typing:

```
$ rvm install 2.5.5
```

This will take between 2 and 10 minutes to run, depending on the speed of your system. It downloads the relevant Ruby source, compiles it and then installs it under ~/.rvm. It will not affect the current version of Ruby which you have installed.

8.2 The upgrade

You will switch to using this new version of Ruby as part of the process of upgrading the Scheduler software.

As a first step, shut down Nginx. This will prevent access to your system for the duration of the upgrade.

```
$ sudo service nginx stop
```

We can then upgrade the Scheduler software in the usual way.

```
$ cd
$ . etc/whichsystem
$ cd $SCHEDULER_DIR
$ git pull
$ cd ..
$ cd scheduler
```

Note those last two steps which are unusual and quite important. In getting the new version of software the file `.ruby-version` in the Scheduler directory will have been changed. By moving out of that directory and then back into it we get RVM to notice the change. You will see a message from RVM saying it is switching to the new version of Ruby.

Next we need to install all the ancillary packages used by Scheduler. This is done using a utility called Bundler, which needs to be installed first.

```
$ gem install bundler --version 1.16.6
$ bundle install
```

The latter command will take a couple of minutes to run, as it also does a bit of compilation.

Then do the usual steps for an application software upgrade:

```
$ rake db:migrate
$ rake assets:precompile
```

and one extra one to set up our new version of Ruby to be used by background jobs:

```
$ rvm alias create scheduler ruby-2.5.5@scheduler
```

Finally you need to edit the file `/etc/nginx/sites-available/scheduler` and change the line:

```
passenger_ruby /home/scheduler/.rvm/gems/ruby-2.3.6@scheduler/wrappers/ruby;
```

to read instead:

```
passenger_ruby /home/scheduler/.rvm/gems/ruby-2.5.5@scheduler/wrappers/ruby;
```

It’s just the version number which you’re changing there - don’t change the path if you happen to be using a user called something other than “scheduler”.

This tells Passenger where to find the right Ruby for your instance of Scheduler.

Finally, you can re-start Scheduler with:

```
$ sudo service nginx start
```

CHAPTER

NINE

UPGRADING DEBIAN

How to upgrade the underlying operating system - Debian GNU/Linux. This section is written to deal specifically with the upgrade from Debian 8 (Jessie) to Debian 9 (Stretch) because that involves some particular issues, but it should be useful for any such upgrade.

Debian GNU/Linux is famed for its stability - once a release is issued it's changed only to fix security issues and it lasts a good long time. This makes it ideal for running servers which need to Just Work(TM).

However, even Debian releases eventually go out of date and stop receiving security fixes so the point comes where one needs to upgrade. Debian Jessie is due to reach end of life on 30th June, 2020. At the time of writing that's more than a year away, but it does no harm to plan ahead.

9.1 Background

Upgrading Debian is generally pretty simple. You edit /etc/apt/sources.list (and all the files under /etc/apt/sources.list.d) changing the name of the target release to the one that you want, then do:

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get dist-upgrade
```

Then re-boot and you're done, *but*. . .

There are some small extra steps needed to make sure your Scheduler installation still works.

Note: You may be interested to check out the full Debian upgrade instructions

The two things which make your Scheduler system upgrade slightly more complicated are:

- MySql
- Nginx + Passenger

Up to and including release 8 (Jessie) the Debian distribution included the MySql database system as one of its packages. With the release of version 9 (Stretch) they decided to include MariaDB (a fork of MySql) instead. Rather unfortunately they chose to name the packages containing MariaDB as if they were direct replacements for MySql. They are called "mysql-server", etc.

Thus, if you do a plain upgrade your MySql installation will be ripped out and replaced with MariaDB. The latter is meant to be a compatible replacement, but initial testing quickly threw up a number of incompatible changes which mean that it isn't.

To keep Scheduler running properly you need to keep MySql on your system, and the following instructions tell you how to do this.

Between Debian 8 and Debian 9 there was also a slight change to how Nginx + Passenger are installed.

In Debian 8, you need to install a special version of Nginx, packaged by Phusion (the makers of Passenger) and compiled to include Passenger.

In Debian 9, this is no longer necessary - you can install the standard Nginx packages from the Debian distribution, then add Phusion Passenger as a plug-in module. This is much tidier, but means an extra step is needed to undo the way in which it was installed originally.

9.2 Step by step

The first thing to check is that you have enough free disk space for the upgrade. 5G free should be plenty.

If you are running your Scheduler installation as some kind of Virtual Machine, you might want to take a snapshot of its state before you start. That way you have an easy disaster recovery path if things go horribly wrong. In any case, make sure you have a good backup of your database and system.

You will need to have root access to your system, or at least full access to sudo. All the following assumes you are logged in as an ordinary user with sudo access.

The whole process will take something of the order of 1-2 hours depending on the speed of your system and Internet connection. It's something to do at the weekend, or during a school holiday.

- Step 1 - Shut down Nginx

  We need to stop access to Scheduler completely, and in any case we're going to do interesting things to Nginx which would stop things from working.

  ```
  $ sudo service nginx stop
  ```

  This will stop both the Nginx web server, and the Passenger processes which run Scheduler.

- Step 2 - Uninstall Nginx and Passenger

  We can now uninstall both Nginx and Passenger. Don't worry - this won't lose any of your individual configuration.

  ```
  $ sudo apt-get remove nginx-extras nginx-common nginx passenger passenger-dev␣
  ↪passenger-doc
  ```

- Step 3 - Switch to using non-Debian MySql packages

  The makers of MySql provide packages explicitly designed to work on Debian - both Debian 8 and Debian 9. We can install those to replace the ones provided by Debian.

  ```
  $ wget https://dev.mysql.com/get/mysql-apt-config_0.8.12-1_all.deb
  $ sudo dpkg -i mysql-apt-config_0.8.12-1_all.deb
  ```

  At this point you'll get a dialogue asking you which version of MySql you want. Accept the suggested version - 5.6.

  ```
  $ sudo apt-get update
  $ sudo apt-get dist-upgrade
  ```

  Because you already have the Debian MySql packages installed, the last command will upgrade them to the ones provided by MySql themselves.

- Step 4 - Upgrade the Debian GNU/Linux operating system

  This is the process mentioned briefly at the top of this page.

Edit `/etc/apt/sources.list` and all the files under `/etc/apt/sources.list.d`. There should be two in the latter directory - one for MySql and one for Passenger. Change all occurrences of "jessie" to "stretch".

In vim you can do this with:

```
:g/jessie/s//stretch/g
```

Then do the actual upgrade with:

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get dist-upgrade
$ sudo apt-get clean
```

Those middle two can take half an hour to an hour each, and have the annoying trick of stopping after a while to ask you a question.

Each of them will first download all the needed files, then start the upgrade. The main point for asking for input is between these two steps, but it can happen at any point in the upgrade.

Note: A common message you will get during the upgrade process is one about the installer wanting to overwrite a configuration file with a newer version. Usually it just goes ahead and does this, but if it thinks it detects that you have modified the current file then it will stop and ask you whether it can overwrite it.

You can look at differences between the two before making your decision, but there is no option to edit or merge them at that point.

Whichever way you decide it will keep a copy of the other one. That is, if you choose to keep the old one, it will put the new one alongside it with a "new" suffix to its name, whilst if you choose to overwrite it will keep the old one with an "old" suffix.

An effective policy is to accept all the new configuration files, but keep a list of them as you go along. Then go in afterwards and edit the new files to re-install your local changes. That way you get any new configuration defaults too. It tends to be less work to put your changes in the new files, than to put all the new bits into your old files.

Once those steps have completed you can re-boot your system and it should then be running Debian 9 (Stretch).

- Step 5 - Temporarily disable Scheduler's Nginx configuration

  We're about to re-install Nginx, but when it first starts it won't have the Passenger plugin. Since Scheduler uses Passenger, the Scheduler configuration file will prevent Nginx from starting.

  The actual Scheduler configuration file exists as `/etc/nginx/sites-available/scheduler`, and then there is a link to it in `/etc/nginx/sites-enabled`. Delete the link with the following command:

  ```
  $ sudo rm /etc/nginx/sites-enabled/scheduler
  ```

  The original file will still exist in `/etc/nginx/sites-available`.

- Step 6 - Install Nginx

  ```
  $ sudo apt install nginx
  ```

- Step 7 - Add Phusion Passenger

  ```
  $ sudo apt install libnginx-mod-http-passenger
  ```

- Step 8 - Re-enable Scheduler

```
$ sudo ln -s /etc/nginx/sites-available/scheduler /etc/nginx/sites-enabled
```

- Step 9 - Re-start Nginx

```
$ sudo service nginx restart
```

And you should be there!

Note: If you start with a really old installation, it’s just possible that your Ruby interpreter won’t work after the upgrade because the libraries with which it was built are no longer there.

If you have this problem you can re-build Ruby really easily with RVM.

```
$ rvm reinstall 2.3.6
```

will get things going again. (Obviously, change the version number to whichever version of Ruby you are currently using.)

CHAPTER TEN

CONTACT

For further information about Scheduler, please contact info@xronos.uk

CHAPTER ELEVEN

AVAILABLE DOCUMENTS

- Scheduler Admin Guide
- Scheduler Advanced User Guide
- Scheduler API Guide
- Scheduler Installation Guide
- Scheduler User Guide

Scheduler is licensed under the GNU General Public Licence, version 2.

Note - Scheduler is *not* a timetabling program. It has no facilities at all to solve the problem of school timetabling. If you want a good timetabling program, try Keith Johnson's Timetabler.

CHAPTER

TWELVE

INDICES AND TABLES

- search