Per-student generation with existing groups

Wise Timetable — student-level checking when the groups are already there

◆ ◆ ◆

Wise Technologies d.o.o. · Wise Timetable desktop application · August 2026

Applies to the 2026 release and later.

Contents

1 What this document covers

There are two ways to make Wise Timetable schedule around the individual student rather than around the group, and they start from opposite ends.

The companion manual, *Per-student timetable generation*, describes the first one. You have no groups at all. You hand the program a teaching plan and an enrolment list as two flat CSV files, it invents a group for every student, merges those groups into turnuses course by course, and the timetable it builds is correct for each person because each person was a unit of the calculation. That route needs two import definitions and, in practice, a pair of dictionary files to reconcile the names your own system uses with the names in the timetable.

This document describes the second one, and it is the situation most institutions are actually in. **The groups already exist. Every student is already in one.** Nothing needs to be imported through a dictionary, no group is created for you, and the structure of your timetable does not change at all. Two checkboxes in `Settings ▸ Miscellaneous ▸ Generate` are the whole feature:

- **Check individual student - classes**
- **Check individual student - groups**

Tick them, and generation stops treating a group as the smallest indivisible unit of teaching. It starts asking, for every pair of lessons it considers putting in the same hour, whether any one person would have to be in both places.

2 The problem it solves

A group is a convenient approximation. It is exact only while every student in it attends exactly the same lessons, and there are four ordinary situations in which that stops being true:

- **Electives.** Twelve students out of group BDE-1 and nine out of BDE-2 take *History of Typography*. They are put into an elective group of their own. To a group-level check, that elective group and BDE-1 are unrelated names, so the generator is free to schedule them in the same hour.
- **Repeat students.** A student retaking one course from the previous year belongs to this year's group and to one course of last year's. No group name records that.
- **Joint teaching.** Two programmes are taught one course together, in one turnus, but split for everything else.
- **Subgroups.** A laboratory group is a child of a lecture group. Its students are in both.

In each case the group structure is right and the timetable built from it is still wrong for a particular person, who finds two lessons in the same hour on their own timetable and reports it after publication. The two checks in this document are how you find that before publication rather than after.

They cost something, and section 9 is honest about what. Nothing here is free: switching them on makes the generator's job strictly harder, so a timetable that was tight before may come back with more unallocated turns. That is the correct outcome. Those turns were never really placeable; the group approximation was hiding it.

3 The settings

Everything is in `Settings ▸ Miscellaneous`. Two tabs are involved.

3.1 Generate tab

English	Slovenian	What it does
Check individual student - classes	Preverjaj povezavo študentov s predmeti	Generation compares the enrolment lists of two lessons
Check individual student - groups	Preverjaj povezavo študentov s skupinami	Generation compares the student membership of the groups attached to two lessons

3.2 General tab

English	Slovenian	What it does
Check students when edit	Preverjaj studente pri urejanju	Applies the same student-level rule while you edit, not only while you generate
Do not load/save students from database	Ne nalagaj/shranjuj študente v bazo podatkov	Skips students entirely. **This must be clear** for anything in this document to work

3.3 Where the settings are kept

This distinction matters when several people work on the same timetable:

- **The two Generate checkboxes and *Check students when edit* are stored in the timetable file itself.** They travel with the document, so a colleague who opens the same file generates under the same rule, and a timetable generated last year still records how it was generated.
- ***Do not load/save students from database* is stored per workstation**, in `wtt.settings`. It is a property of the machine, not of the timetable, so it has to be checked on every computer that opens the file.

4 What each check actually compares

The two checkboxes are not two strengths of the same test. They read two different pieces of data, and an institution that has one of them and not the other should tick only the one it can support.

4.1 Check individual student - classes

For every lesson the generator is about to place, it builds the list of students who are **enrolled in that course with that execution type** - the same course and the same *Lecture* / *Seminar* / *Laboratory* line. Group membership is not consulted at all. Two lessons are then forbidden to overlap if those two lists share even one student.

The consequence worth understanding: **this check works on a course that has no groups attached to it whatsoever.** If your enrolment data is authoritative and your groups are decorative, this is the check you want.

Where the enrolment comes from: the student's own course list, `Edit ▸ Students ▸ Courses` (*Student's Courses* / *Predmetnik*), filled in by hand, by the students CSV import, or from your student information system.

4.2 Check individual student - groups

For every lesson, the generator instead builds the list of students who are **in the groups allocated to that turn**. A student counts if one of their groups is the allocated group itself, its parent group, or one of its subgroups - so a lecture group and its laboratory subgroup correctly share their students.

The test is then applied between two lessons only when the groups differ; a lesson is never made to clash with another lesson of its own group, because that case is already handled by the ordinary group rule that has always been there.

If your enrolment lists are thin or absent but your group membership is real, this is the check that pays.

4.3 Side by side

	Check … - classes	Check … - groups
Reads	course enrolment of each student	group membership of each student
Needs groups on the turn	no	yes
Catches the elective taken across two groups	yes, if the enrolment says so	yes, if the elective group holds the students
Catches the repeat student	yes	only if they are in both groups
Understands subgroups	not applicable	yes - parent and child count as one
Blind spot	a student whose enrolment list is empty	a student in no group

Most institutions tick both. They overlap heavily but they fail differently, and the union of the two is what a person's real week looks like.

5 Before you start

Five things have to be true. Four of them are about your data and one is a setting.

1. **The students are in the document.** Open `Edit ▸ Students`. If the list is empty, nothing in this manual has anything to work on and generation behaves exactly as before - silently, with no warning.
2. `Do not load/save students from database` **is clear** on the workstation you are generating on (`Settings ▸ Miscellaneous ▸ General`). With it ticked the students are never loaded, so the lists the two checks build are all empty.
3. **Group membership exists**, if you intend to use *Check individual student - groups*. The Groups screen shows *Unallocated* for each subject area; that figure should be zero, or you should know why it is not.
4. **Course enrolment exists**, if you intend to use *Check individual student - classes*. A student with an empty course list is invisible to that check.
5. **Every group attached to a turn still exists.** A turn allocated to a group that has since been deleted is dropped from the generation with a line in the log; see section 10.

6 Getting students and their groups into the file

If your students are already there with their groups, skip to section 7. This section is for the first time.

6.1 By hand

`Edit ▸ Students` (*Vnos sprememb ▸ Študenti*) lists everyone, filtered by programme, year and subject area. Select one row or drag a block of rows, then `> Group` (*> Skupino*) opens the *Adding Students* dialog and puts the selection into the group you choose. *Show Only Available* hides students who already have a group, and *Adjust Group Size* changes the group's headcount to match what you actually put in it.

`Courses` (*Predmeti*) on the same screen opens the student's own course list - the enrolment that *Check individual student - classes* reads.

6.2 Automatically, from the group sizes

On the Groups screen (`Edit ▸ Groups`), for one subject area at a time:

- `Distribute Students` (*Razporedi študente*) spreads the students of this subject area who are not yet in a group over the groups that exist, filling them to their stated sizes.
- `Distribute All Students` (*Razporedi vse študente*) does the same for every subject area at once.
- `Release Groups` (*Sprosti skupine*) and `Release All Groups` (*Sprosti vse skupine*) empty them again. They do not delete anybody - the students become unallocated.

Both distribute buttons ask you to confirm first, because the pending changes to the groups themselves are applied before the students are spread.

6.3 From a CSV file

`Data ▸ Import Data From CSV File ▸ Import Students`. There are two layouts, and which one is read depends on a setting.

The standard layout, semicolon separated, one student per line:

```
ID;Year;Surname;Name;Email;Group;LDAP name;COURSE CODE;COURSE CODE;...
```

The first three fields are required. `Group` is a single group, matched against the group's name exactly. The course codes from the eighth field onwards are the enrolment, and they are what *Check individual student - classes* will read. A line whose first field contains `ID` is treated as a header and skipped; `delete` in the name field removes that student.

The alternative layout is the one to use when a student is in several groups at once. Tick `Settings ▸ Miscellaneous ▸ General ▸ Use alternative import/export from CSV files` (*Uporabi alternativni uvoz iz CSV datotek*) first. Then:

```
ID;Name;Surname;Group 1;Group 2;Group 3;...
```

Up to fifteen group columns are read, each matched against a group name exactly, and the student is put into every one that matches. A field that looks like an e-mail address is stored as the e-mail instead, and a first group column holding a single digit 1-9 is read as the year of study.

Two things about re-importing. By default a second import of the same student ID **replaces** their group list rather than adding to it; set `StudentImportMergeGroups=1` in the `[ADVANCED]` section of `wtt.settings` if you want the imports to accumulate. And an export from your own system is usually still in the machine's ANSI code page rather than UTF-8 - Wise Timetable detects that and converts it on the way in, recording the conversion in the log, so you do not have to prepare the file.

6.4 From the database

`Data ▸ Import From Database ▸ Import Students` reads them from the institutional database your licence is configured for. This is the route to prefer where it exists: the enrolment is then whatever the registrar's office says it is on the day you generate.

7 The procedure, step by step

7.1 Check that the data is complete

Open the Groups screen and read *Unallocated* (*Nerazporejenih*) for each subject area. Then open `Edit ▸ Students` and check the count. These two numbers are the whole precondition; if they are wrong, every step below runs and produces a timetable that looks fine and checks nothing.

7.2 Save an empty version first

Before the first generation, save the file with no lessons placed. Every subsequent run can then start from a known state, which matters more here than usual: switching a student check on changes what the generator considers legal, so the run after the change is not comparable with the run before it.

7.3 Turn the checks on

`Settings ▸ Miscellaneous ▸ Generate`, tick *Check individual student - classes*, *Check individual student - groups*, or both, following section 4. On the `General` tab, tick *Check students when edit* as well if you want the same rule to apply while you drag lessons afterwards - see section 8.

Confirm the dialog. The settings are now part of the document, so save it.

7.4 Lock what must not move

Anything already placed correctly - senate slots, part-time teaching, external lecturers, anything booked by hand - should be locked before you generate, exactly as in an ordinary run. Locked lessons are never touched, and the student checks apply to everything the generator places around them.

7.5 Generate

Run generation as usual, choosing the programmes and years. Watch the count of unallocated turns at the end and compare it with the same run before the checks were on. An increase is expected and is the point of the exercise; a large increase says the group structure and the enrolment data disagree with each other, and section 10 is where to look.

7.6 Read the result

Two places show what the checks found:

- `Analysis ▸ View Conflicts` has a legend entry for **Student Overlaps**, alongside room, lecturer and group overlaps. This is where a clash that only one person can see is reported.
- `Analysis ▸ Display Unallocated Groups` lists what could not be placed at all.

Then look at one real student. Set the main selector on the workspace to **Students** and pick one: it shows that individual's whole week, assembled from every group and elective they are enrolled in. It is the view a student sees, and it is the one to check before publishing.

8 After generation: the same rule while you edit

Generation is one thing; the hundred manual corrections afterwards are another, and by default they are checked at group level only. `Check students when edit` (*Preverjaj studente pri urejanju*) extends the rule to them.

What it does is worth stating precisely, because it is not quite the same computation as the generator's. Wise Timetable walks the student list and, for every student in two or more groups, records that **those groups clash with each other**. The result is a per-group list of other groups that share at least one person, and that list is then consulted wherever a group conflict would be:

- the room and lecturer views, when they decide whether an hour is free;
- the occupancy screen and the reservation dialogs;
- the conflicts screen.

Two consequences follow from the fact that this is a group-level summary of student-level data:

- **It is exactly as good as your group membership.** A student whose groups are right produces the right clashes. Course enrolment plays no part in it - that is the generator's *classes* check, and it has no edit-time equivalent.
- **A group belonging to a mandatory course clashes with every other group in the same subject area.** That is deliberate and predates this feature: a course everybody must take cannot overlap anything on the same subject area.

The setting is stored in the timetable file, so it applies to everyone who opens it.

9 What it costs

Both checks make the generator carry extra data and ask an extra question about every pair of lessons it considers, so it is fair to know the shape of the cost before switching them on across a whole institution.

Memory. The student lists are held in side tables, allocated only when the corresponding checkbox is on. With neither ticked they are not allocated at all and a lesson costs nothing extra. With them ticked, the space is one entry per matching student per lesson for the *classes* check, and one per student per allocated group for the *groups* check.

Time. The question the generator asks is whether two student lists share anybody. It is asked for every element of the week, hundreds of thousands of times in a run, and at an institution that checks students both lists run to hundreds of entries. Since the 2026 release the list on one side is sorted once per call and the other is probed with a binary search, which is what makes the feature usable on a full university file; before that it was a nested scan of both lists and the cost grew as their product.

In practice. Expect a generation that took under a minute to still take under a minute, and a very large file to take noticeably longer. If a run becomes unacceptably slow, the first thing to try is ticking only the check you actually need - section 4.3 says which one that is.

Two limits you will not normally reach. A single turn is generated with at most the number of groups the generator's tables hold; if a turn allocates more, the extra ones are dropped and a warning is written to the log. And the edit-time clash builder considers at most 4096 groups for one student, which is also logged if it is ever hit.

10 Messages and what to do about them

These are written to `C:\ProgramData\WiseTimetable\wtt_event.log`. Read that file first whenever a generation behaves unexpectedly; it carries the exact turn and group numbers.

Message	What it means	What to do
Turn part N is allocated to group G, which no longer exists; it is left out of the generation	A group was deleted after it had been attached to a turn	Open the course, re-attach a live group to that turn. Until you do, the turn is not generated at all
Turn part N allocates M groups; the generator tables hold K, so only the first K are used	One turn has more groups attached than the generator can carry	Split the turn, or reduce the number of groups attached to it
student N is in more than 4096 groups - the rest are not considered for clashes	Almost always an import that accumulated instead of replacing	Check `StudentImportMergeGroups` in `wtt.settings`, then re-import cleanly
Year Y not valid! ... we will ignore it	A students CSV carries a year outside the allowed range	Correct the year column in the export
The document already holds the maximum number of students	The students CSV is larger than the document can take	Import per subject area, or reduce the file

And three symptoms that produce no message at all:

- **Generation is no slower and nothing changed.** The students were never loaded. Check *Do not load/save students from database* on this workstation, and check that `Edit ▸ Students` is not empty.
- **The *classes* check finds nothing.** The students have no course lists. That check reads enrolment, not groups.
- **The *groups* check finds nothing.** The students are unallocated. The Groups screen shows how many.

11 Checklist

- ☐ Students are in the document, and the count is what you expect
- ☐ *Do not load/save students from database* is clear on this workstation
- ☐ Unallocated students per subject area is zero, or explained
- ☐ Course enrolment is present, if you are using the *classes* check
- ☐ An empty version of the file is saved before the first run
- ☐ *Check individual student - classes* / *- groups* ticked as decided in section 4
- ☐ *Check students when edit* ticked if manual corrections should follow the same rule
- ☐ The file is saved after changing the settings, so colleagues inherit them
- ☐ Everything that must not move is locked
- ☐ After the run: unallocated turns reviewed, Student Overlaps checked on the conflicts screen
- ☐ One real student's own week checked in `Workspace ▸ Students` before publishing

Appendix A - A demonstration you can build in ten minutes

The point of this appendix is to see the two checks disagree, on a file small enough to hold in your head. It uses the same fictional institution as the companion manual, the **Wise University of Design and Media**, and 24 students of one programme.

The accompanying files are `WTT-groups-demo-students.csv` and `WTT-groups-demo-enrolment.csv`.

A.1 The structure

One programme, **BDE** (*Bachelor of Design*), year 1, one subject area, **Design**. Five groups:

Group	Parent group	Students	Who
BDE-1	-	12	20260001 - 20260012
BDE-2	-	12	20260013 - 20260024
BDE-1a	BDE-1	6	20260001 - 20260006
BDE-1b	BDE-1	6	20260007 - 20260012
TYP-E	-	8	20260003, 20260004, 20260005, 20260011, 20260012, 20260013, 20260018, 20260022

`TYP-E` is the elective group, and it is the interesting one: five of its members come from BDE-1 and three from BDE-2, so its name tells you nothing about which other group it overlaps.

Three courses:

Course	Execution type	Groups attached to the turn
Design Studio	Lecture	BDE-1, BDE-2 (one turnus)
Design Studio	Laboratory	BDE-1a, BDE-1b, BDE-2 (three turnuses)
History of Typography	Lecture	**none**

Leaving *History of Typography* with no group attached is deliberate. It is how an elective that comes out of a student information system usually arrives - as an enrolment list, before anybody has made a group for it.

A.2 Building it

1. Create the programme, the subject area and the five groups on the Groups screen, setting the parent group of BDE-1a and BDE-1b to BDE-1.
2. Create two execution types and **give them codes**: *Lecture* with code `L`, *Laboratory* with code `LAB`. The enrolment file names them by code, so this step is not optional - an enrolment row whose type code matches nothing is stored against no execution type at all, and the *classes* check then never matches it to a lesson.
3. Create the three courses of the table above, giving each one a **course code**: `DS` for *Design Studio*, `TYP` for *History of Typography*. Attach the groups as shown, and give each turn two hours a week.

4. Tick `Settings ▸ Miscellaneous ▸ General ▸ Use alternative import/export from CSV files`.
5. `Data ▸ Import Data From CSV File ▸ Import Students`, and choose `WTT-groups-demo-students.csv`. Twenty-four students arrive, each already in their groups. The Groups screen should now show **0 unallocated**.
6. Clear the alternative-import setting again, and import `WTT-groups-demo-enrolment.csv` the same way. This is the ordinary layout; its last columns are the enrolment, written as `code(type code)` - `DS(L)`, `DS(LAB)`, `TYP(L)`. Open one student's `Courses` and check that three rows appeared.

Do those two imports in that order. The alternative layout replaces a student's group list, the standard layout does not, so groups-then-enrolment keeps both. The other way round, the second import would empty the groups the first one built.

A.3 What to look for

Generate four times, changing only the two checkboxes, and watch whether *Design Studio - Lecture* and *History of Typography - Lecture* are allowed into the same hour:

Check … - classes	Check … - groups	Result	Why
off	off	they overlap	Nothing connects the two lessons: one has BDE-1 and BDE-2, the other has no group at all
off	on	they still overlap	The *groups* check needs groups on the turn, and *History of Typography* has none
on	off	**they are kept apart**	Eight students are enrolled in both courses, and the enrolment is what this check reads
on	on	kept apart	As above; the *groups* check adds nothing here

Then attach TYP-E to the *History of Typography* turn and repeat. Now the *groups* check catches it too - TYP-E shares five students with BDE-1 and three with BDE-2 - and the two checks agree.

One more thing to try, because it is the case a group-only timetable always gets wrong: schedule *Design Studio - Laboratory* for BDE-1a against *History of Typography*. Students 20260003, 20260004 and 20260005 are in both. With neither check on, the timetable is clean and three people have two lessons in one hour.