Table of Contents

Scope

This document is intended for AAPS users who want to investigate what effect certain settings in AAPS have on the insulin delivery. It describes installation and usage on Android and PC. The last version is heavily biased to support autoISF but the the plain AAPS things still work as in the beginning. If your AAPS dols not include autoISF or you study older logfiles that do not include the buzzwords from autoISF you can still use the emulator by adding some commands at the beginning of the VDF-File as outlined in appendix A.

Any other dialect of AAPS like AIMI, Boost or DynamicISF cannot be used.

How it works in principle

The idea is to reduce risks when changing AAPS settings. I first test the impact of such changes offline on a PC by emulating in a python script what the loop would do. That python is a translation of the original AAPS SMB „determine-basal.js“ javascript version.

The code scans an existing AAPS logfile and extracts the results of the original run. Also the inputs for "determine-basal" like last glucose, IOB array or profile definition are extracted. Here profile is not the usual daily pattern of basals etc. but a collection of settings like which SMB feature is ON or OFF; for details go to the „SMB“ tab in AAPS and scan the section named „Profile“.

After collecting all these inputs the emulator will also read your definition of alternative AAPS settings like

- different temp target
- different SMB settings
- different basal safety factors
- different ISF (here named „sens“ !) or IC values

Settings like those related to meals already eaten or insulin already dosed are handled in other modules of AAPS and are therefore out of reach here. For the same reason changes to Autosense cannot be handled and the only meaningful change you can do is set it to „1“ which means switch it off.

Then determine_basal.py is called to rerun the analysis and produce results in the working folder for comparing the original situation with the emulated scenario.

How to find improved settings at minimal risk

Because this tool is a virtual analysis you can emulate many changes „in silico“ first without risk to your health should those changes turn out to be unfavourable. The steps to undertake with the help of this tool go as follows:

1. Identify a situation you want to improve and get the relevant logfiles
2. Read the related section of the docs
3. List the settings that may have an impact
4. Use this emulator on the PC to try which of those settings have the desired impact
5. Tune the level of the favourite setting to deliver the desired improvement
6. Rerun the emulator on the AAPS phone so you see the result quasi in real real time and fine tune the setting. Here use the variant with the lower insulin delivery as the master in AAPS and the other one as the emulated version. This way you can decide each time an incremental insulin dose is suggested whether you really want to apply it manually like you did in Open Loop while learning AAPS.
7. Once things look OK and safe implement the setting in AAPS itself

Steps 4-6 are those not available without this tool and add extra safety when changing settings.

Comments and current limitations

This is is not a complete virtual loop! Please keep in mind that the variant shows a new and independent decision at each time-step. It is not a sequence of decisions building on each other because there is no forecasting what a different decision would have lead to at the next time step. Therefore you need to carefully consider the results at each time-step separately and consider whether it is safe enough to be activated in AAPS. Once this emulator gives me promising hints and the parameter is tuned I go to quasi "open loop", e.g. for a few days I manually apply an increase in SMB as a correction bolus before going live.

Be aware that maxBolus=0 in the tabular output file means maxBolus was high enough to not limit the SMB and was therefore not listed explicitly in the original logfile by AAPS.

Glucose values are assumed to be in mg/dl.

The code only works for oref1 in SMB mode even when SMB is not activated.

Tested system settings

- Windows10, 64 bit
- python 3.7, 3.8, 3.10
- some python libs like matplotlib (see the import statements in vary_settings_core.py)
- AAPS versions for logfile: oref1 & SMB for 2.3 – 3.0.0.1
- AAPS versions for determine_basal: oref1 & SMB for 2.5 – 3.0.0.1

Installation

Download the Github repo (https://github.com/ga-zelle/APS-what-if) as a zip archive.

On Windows10

Extract it to a folder containing

- "determine_basal.py" as the emulation of the original java version
- "vary_settings_GUI.py" which manage the user input in an interactive window
- create a shortcut on your desktop to point to this GUI file.
- "vary_settings_batch.py" which alternatively manages the he user input in a DOS window
- "vary_settings_core.py" the main programme which manages the whole process
- "<variant_label>.vdf" which contains the definition of the settings you want to change

To get python if you do not have it yet on your PC the best option is to download its installer from https://realpython.com/installing-python/#how-to-install-python-on-windows. When the installer starts select user specific options rather than standard configuration. In the dialogue that follows select

- add python to the system variable for PATH
- select to also download and install “pip”

After the python installation is completed you need to download the matplotlib library. This is done in a DOS window by entering ”pip install matplotlib” in the command line.

The last action to make python fit for this project is to define a system variable PYTHONUTF8=1 to allow use of special characters like the Greek Δ from the UTF-8 character set.

On the AAPS phone

- From the play store download and install qpthon3 („QPython 3L – Python for Android“ by QpythonLab).
- This should create a folder „qpython“ at the top level
- Go to its subfolder „scripts3“
- From the above Github archive downloaded above extract the following script files (i.e. the python programmes) to this subfolder
  - „determine_basal.py“
  - „vary_settings_batch.py“
  - „vary_settings_core.py“
- "<variant_label>.vdf , the VDF (see below), is the same as on Windows and is copied to the same folder containing the AAPS logfiles.

Define your variant in the VDF (Variant Definition File)

Create or edit "<variant_label>.vdf", your VDF. The previous naming convention "<variant_label>.dat" is still supported but discouraged. It defines the changes for your what-if scenario. You have access to the relevant lists or dictionaries as shown in the tab „SMB“, sections „Glucose-data“ through to „Autosense-data“. The general format per row is:

```
.
.
<array>   <item in array>  <new value or formula>    [<optionally: ### any comment>]
.
.
```

The entries may be separated by several BLANKs or TABs and aligned for better readability. Their meaning is as follows:

- *<array>*
  name of python dictionary; the available names are

autosens_data	do not change; gets calculated inside AAPS elsewhere
glucose_status	not really meaningful
currenttemp	not really meaningful
iob_data	do not change; gets calculated inside AAPS elsewhere
meal_data	do not change; gets calculated inside AAPS elsewhere
profile	most variations will happen here
new_parameter	to hold emulator specific information like AAPS version

  If array is not a recognised name the row will be skipped.
  No BLANKS allowed here.
- *<item in array>*
  Item in <array>; may be new or for redefining an existing item in that array
  No BLANKS allowed here.
- *<new value or formula>*
  Boolean, numeric or string value or expression that evaluates accordingly
  May contain BLANKS but not '###'

optionally also:

- *<### any comment>*   an optional comment after a block of '###'

The programme checks the syntax of the formulae entered as not everyone is familiar with python syntax used here. If for example you forget the quotes for max_bg like here:

```
new_parameter   LessSMBbelow          profile[max_bg] + 10      ### ... bg below this value
```

you will get a warning message like below and the programme will exit:

```
*******
Problem in VDF-file in row reading
new_parameter   LessSMBbelow          profile[max_bg] + 10      ### ... bg below this value


error message is:<class 'NameError'>
*******
```

For detailed content of the arrays and their correct spelling see the original logfile and look for rows containing "[DetermineBasalAdapterSMBJS.invoke():" or check the appropriate sections in the AAPS “SMB“ tab.

> **CAUTION:**
> Whereas AAPS will check for limits and deactivate some SMB settings this emulator does not know about them and just goes ahead. Therefore it is your responsibility to check whether AAPS would accept the settings from your variant definition file.

Example: This repo contains the variant definition file „Demo_Sports_Adaptations.vdf“. It will be used for a logfile covering 2-3 hours of playing badminton. I like to stay below 140mg/dl because otherwise I lose concentration and react too slow. In this case it worked quite well but what if I tweak things a bit?

```
profile     enable_autoISF                False                                       ### for logs before autoISF
profile     smb_delivery_ratio            0.5                                         ### AAPS default

profile     min_bg                        101                                     ### just above the temp target threshold for SMB
profile     max_bg                        profile['min_bg']                           ### redefine target
#rofile     target_bg                     int((profile['min_bg'] + profile['max_bg'])/2)  ### example of numerical expression
profile     tempTargetset                 True                                        ### spelling error
profile     temptargetSet                 True                                        ### i.e. not using NS profile target
profile     allowSMB_with_high_temptarget False                                       ### was True at the time
```

1. The first two rows are only required for older logfiles that where created before autoISF was included in AAPS.
2. The 3rd row changes the lower target from 100 to 101 to move it just outside of the „enableSMB_with_temptarget“ range.
3. The 4th row sets the upper target value the same as the lower one with a simple expression. It needs to be the same because I want to simulate setting a temp target which does not allow for a finite bandwidth if defined in the AAPS GUI.
4. The 5th row is ignored because „#rofile“ is not a recognised dictionary. But just look at the rather complex expression assignment possible. As far as targets are concerned the „target_bg“ is defined as the average of „min_bg“ and „max_bg“ anyway inside determine_basal meaning this row is not required anyway.
5. The 6th row tries to redefine the flag for „tempTargetset“ as True. However, that spelling is not really correct because there are 2 swaps in lower case and upper case spelling.
6. The 7th row keeps the flag for „temptargetSet“ as True and this time in correct spelling. There is no check to verify the target bandwidth is zero in such a case.
7. Finally the 8th row disables SMB for higher TempTargets, i.e. above 100.

In summary the target increases from 100 to 101 which is now in line with the general recommendation for sport by disabling SMB.

For first time use I recommend to leave things unchanged and verify that the results agree with the original. Minor numerical deviations are probably due to rounding in the original logfile.

There are six special cases of quasi arrays that were introduced to handle interim or time varying assignments:

- temp   <var_name>   <value or expression>
  You can use your own names for variables to hold interim values which can be

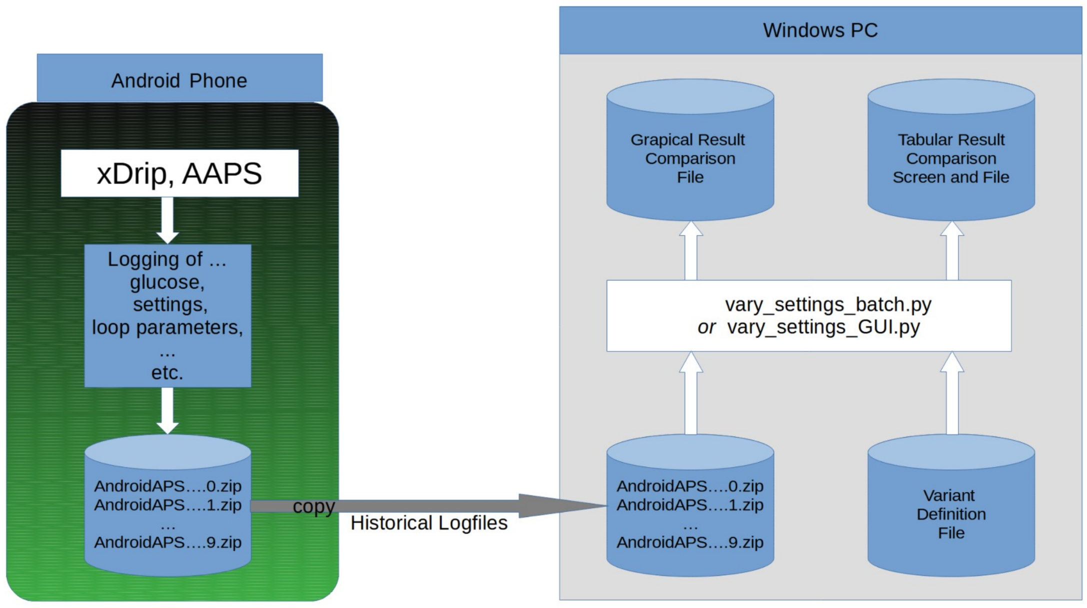
	referenced in statements following it	
• STAIR	<UTC-date/time>	< value from this time onwards>
	This is useful for time dependant values like target switches	
• STAIR_BAS		define pump profiles; see appendix STAIR ...
• STAIR_CR		… for details ...
• STAIR_ISF		… and example
• INTERPOL	<UTC-date/time 1>	< value at this time>
INTERPOL	<UTC-date/time 2>	< value at this time>
	This uses linear interpolation and extrapolation to define a time dependent function	

An example shows possible usage of how to define and how to reference the functions:

```
STAIR           2020-04-14T00:00:00Z          111
STAIR           2020-04-14T12:50:00Z          125
STAIR           2020-04-14T13:10:00Z          131
INTERPOL        2020-04-14T13                 100
INTERPOL        2020-04-14T13:30:00Z          140

profile         min_bg                        round(INTERPOL) - 20          ### time varying value
profile         max_bg                        STAIR - 10                    ### time varying value
```

Execute the analysis on a PC running Windows

For those who are less fit in DOS commands I created a GUI front end which collects the inputs in a form (see next section). Here is a sketch of the data and process flows when using the emulator on a PC running windows:



Sketch of Running the Emulator on a Windows PC

To start, open a DOS command window, navigate to the working folder containing "<variant_label>.vdf" and enter
DOS-prompt>[python] vary_settings_batch.py <logfile(s)> <options> <VDF> [<from> [<to>]]
The initial „python“ can be omitted if the DOS path definition includes the python executable folder.

The meaning of the arguments is as follows:

- *<logfile(s)>*
  the AAPS logfile (still zipped or unzipped) downloaded from Android.
  Optionally you may use the wild characters (? for matching any single character; * for matching any string) to match several logfiles in the directory which provides a longer time window to evaluate. An example could be „AndroidAPS._2020-04-02_*“ to handle all logfiles of that day.

- *<options>*
  "All" or a slash separated list of things to be shown so it does not overload the plot.
  Things to be shown are items from this list:

insReq	is the total insulin required at the current time step
maxBolus	is the upper limit of allowed SMB
SMB	is the actual SMB to be delivered
basal	is the actual basal insulin to be delivered
target	is the lower and upper target
bg	is the original glucose value
cob	is the original COB value to help understanding the situation
iob	is the original IOB value to help understanding the situation
activity	is the original insulin activity to help understanding the situation
as_ratio	is the Autosense ratio
autoISF[1]	is the final ISF factor
ISF-factors[1]	are the individual ISF factors (only tabular outputs, not plotted)
ISF	is the insulin sensitivity factor
fitsParabola[1]	series of fitting a parabola through last BG values with good
correlation and plot them in grey as dotted curve including 5 minute forecast	
bestParabola[1]	out of all those fits show the best one as black curve
range[2]	parameters of nearly unchanged BG from which dura_ISF is determined
slope[2]	parameters of nearly linear change in BG and plot of that line
fitsslope[2]	series of fitting a straight line through last BG values with good
correlation and plot them in grey as dotted line	
bestslope[2]	out of all those fits show the best one as black line
pred	plots the prediction curves per time-step of the emulated scenario including hints if SMB needed to be disabled; you get a quasi animation by using forward/backward keys in the pdf-file created with one page per loop execution
flowchart	in the graphics output only it plots a flowchart of how determine-basal was transversed with all the if-branches executed.

  With the special flag „-“ in front of any option you can suppress individual <plot-options>. I normally use „All/-pred/-flowchart“ which results in everything apart from „pred“ and „flowchart“ in order to speed up the programme significantly in routine analyses.
- *<variant_label>* or alternatively*:*
- *<variant_label>.vdf* (or for downward compatibility *<variant_label>.dat)*
  A label for the result file names and plot. It is also the base name of the VDF "<variant_label>.vdf".
  The alternative method of giving the filename directly may be easier especially when using the DOS type ahead method

The optional parameters may be used to define a time window to be evaluated, whether for a single logfile or a list of them. Please be aware that the date/time combination is given in UTC format, i.e. in Greenwich time without daylight savings effect:
- <from>
  Define the beginning of the time window like „2020-04-02T22:00:00Z“ or for short „2020-04-02T22“ which is midnight of April 2. If you want to skip the start time then just use „2000“ as a very early year.
- <to>
  Optionally also define the end of the time window, again in the readable UTC format used in the logfile system.

The process will create the following files:
- *<logfile>.<variant>.log*

1 For details and further related parameters see https://github.com/ga-zelle/AndroidAPS/tree/autoISF
2 Experimental feature; can be used to test advanced adjustments like ISF depending on slope

echo of script versions used and command line arguments listed just above
echo of parameter changes as requested by <variant>.vdf

- *<logfile>.orig.txt*
  extract of original logfile with entries relevant for the loop result; for the same logfile this will always be the same name and be independent of the variant; therefore there is no harm when a different variant is emulated and it gets overwritten.
- *<logfile>.<variant>.txt*
  equivalent, emulated subset for the variant loop result
- *<logfile>.<variant>.csv*
  comparison of all key values showing original versus emulated run; summary lines at the bottom show minimum and maximum values found within the time window as well as total basal and total SMB delivered
  ready for import into your favourite spreadsheet tool; set import file format to UTF-8 and second column to time format „hh:mm:ss“
- *<logfile>.<variant>.pdf*
  plot of the tabular results; if the previous file with that name is still open in pdf-viewer a warning will be printed until it is clear to proceed.
- *<logfile>.<variant>.delta*
  Experimental table with various sources for glucose delta

A table with the subset of key values as selected by the options list is also output to the screen directly. The plot is saved in a predefined resolution(200dpi) and size(9 x up to 12 inches, portrait). If the emulator was started by this DOS method and without the „pred“ option then the graphic output is also displayed interactively and things like aspect ratio or margins can be modified. After such modifications the graph can be saved manually by clicking the disk symbol, preferably in jpg or png format which makes it easier to later include it in other documents. With the “pred“ option you can inspect it in a pdf-viewer and transverse the history of the logfile with the forward and backward keys, one page per loop execution.

A file compare between the two txt-files is possible, but mainly the formatting is still different and flags too many lines although numerically equivalent. Personally I find the table and plot more useful.

Example

What do the result files look like for the example from above? First let us check „AndroidAPS._2019-11-13_00-00-00_.7.Demo_Sports_Adaptations.log“:

```
===============================================================
Echo of software versions used
-------------------------------------------------------------------
 vary_settings home directory  S:\AAPS3p0_dev\
 dated: 2022-05-06 01:59:23,   module name: vary_settings_GUI.py
 dated: 2022-04-27 02:20:08,   module name: vary_settings_core.py
 dated: 2022-04-24 17:50:29,   module name: determine_basal.py
-------------------------------------------------------------------
Echo of execution parameters used
-------------------------------------------------------------------
Logfile(s) to scan    L:/PID/AndroidAPS._2019-11-13_00-00-00_.7
Output options        All/-pred/-flowchart
Decimal symbol        .
Start of time window  2000-01-01T00:00:00Z (default)
End of time window    2099-12-31T23:59:59Z (default)
===============================================================

========== Echo of what-if definitions actioned for variant Demo_Sports_Adaptations
========== created on Fri, 06 May 2022 02:14:05 +0200
========== for loop events found in logfile L:/PID/AndroidAPS._2019-11-13_00-00-00_.7

loop execution in row=2050 of logfile AndroidAPS._2019-11-13_00-00-00_.7 at= 2019-11-13T18:23:30Z
edited old value of 100 in profile with min_bg=101
edited old value of 100 in profile with max_bg=101
not actioned: [#rofile], [target_bg], [int((profile['min_bg'] + profile['max_bg'])/2)]
appended new entry to profile with tempTargetset=True
```

```
edited old value of True in profile with temptargetSet=True
edited old value of True in profile with allowSMB_with_high_temptarget=False

loop execution in row=5288 of logfile AndroidAPS._2019-11-13_00-00-00_.7 at= 2019-11-13T18:28:30Z
edited old value of 100 in profile with min_bg=101
edited old value of 100 in profile with max_bg=101
not actioned: [#rofile], [target_bg], [int((profile['min_bg'] + profile['max_bg'])/2)]
appended new entry to profile with tempTargetset=True
edited old value of True in profile with temptargetSet=True
edited old value of True in profile with allowSMB_with_high_temptarget=False
…
```

The initial few lines show which software modules were actually used. In the past there were discrepancies between what the tool did and what it should have done. The next few lines list the execution parameters and whether those were default settings.

Now the bulk part starts with an echo of the changes assigned at each time-step. It is useful to check results of complex expressions for values and verify that there were no spelling errors like in the „appended ...“ rows

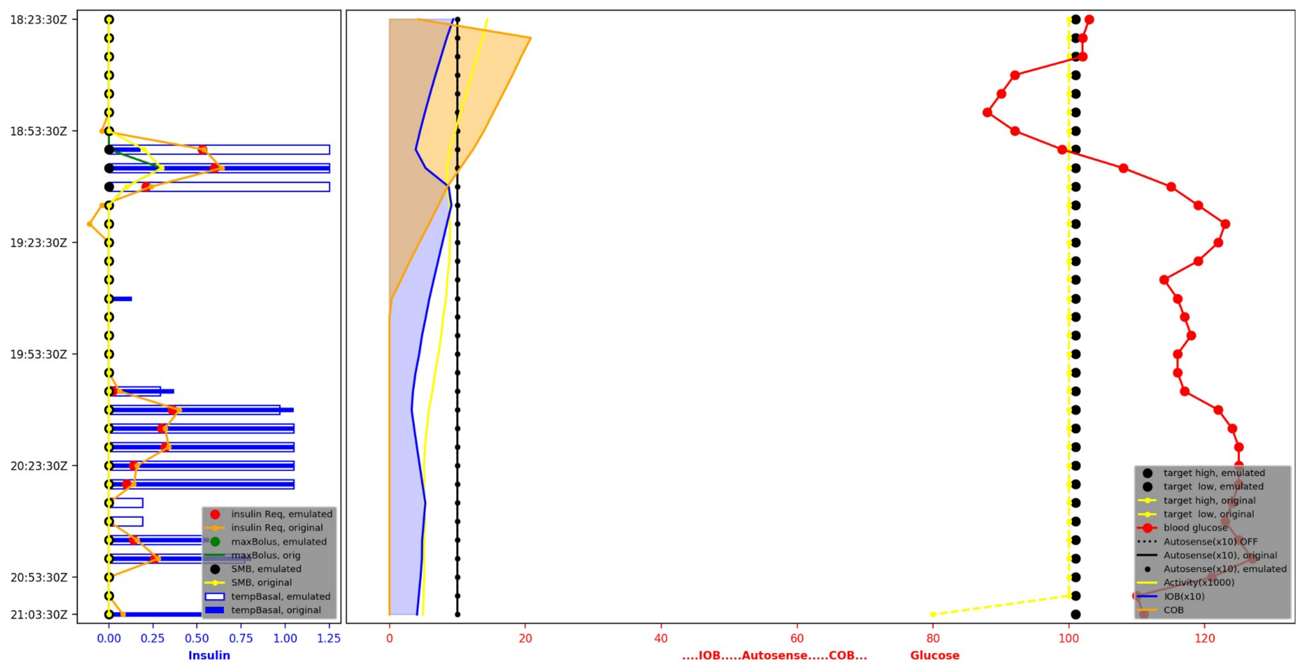
Next check „AndroidAPS._2019-11-13_00-00-00_.7.Demo_Sports_Adaptations.tab“ (please note that the new executable creates a CSV-file instead and holds far more columns than printed here):

```
                                                   -Autosens-   -----target-----    insulin Req     -maxBolus-      ---SMB---      ---tmpBasal---
id    time     --UNIXtime--   bg    cob   iob    act  orig  emul    orig     emul     orig   emul     orig emul      orig emul       orig    emul
 0 18:23:30Z  1573669410.3  103    4.2  0.94  0.014   1.0   1.0  100-100  101-101       0      0        0    0         0    0          0       0
 1 18:28:30Z  1573669710.2  102   20.8  0.84  0.014   1.0   1.0  100-100  101-101       0      0        0    0         0    0          0       0
 2 18:33:30Z  1573670010.3  102   19.4  0.76  0.013   1.0   1.0  100-100  101-101       0      0        0    0         0    0          0       0
 3 18:38:30Z  1573670310.3   92   18.1  0.68  0.012   1.0   1.0  100-100  101-101       0      0        0    0         0    0          0       0
 4 18:43:30Z  1573670610.3   90   16.7   0.6  0.011   1.0   1.0  100-100  101-101       0      0        0    0         0    0          0       0
 5 18:48:30Z  1573670910.3   88   15.3  0.52   0.01   1.0   1.0  100-100  101-101       0      0        0    0         0    0          0       0
 6 18:53:30Z  1573671210.3   92   13.9  0.45   0.01   1.0   1.0  100-100  101-101   -0.04      0        0    0         0    0          0       0
 7 18:58:30Z  1573671510.3   99   12.3  0.38  0.009   1.0   1.0  100-100  101-101    0.54   0.53        0    0       0.2    0       0.18    1.25
 8 19:03:30Z  1573671810.6  108   10.4  0.53  0.009   1.0   1.0  100-100  101-101    0.64    0.6      0.3    0       0.3    0       1.25    1.25
 9 19:08:30Z  1573672110.3  115    8.5  0.87  0.009   1.0   1.0  100-100  101-101    0.24   0.21      0.1    0       0.1    0          0    1.25
10 19:13:30Z  1573672410.3  119    7.2  0.91  0.009   1.0   1.0  100-100  101-101   -0.04      0        0    0         0    0          0       0
11 19:18:30Z  1573672710.3  123    5.8  0.84  0.009   1.0   1.0  100-100  101-101   -0.11      0        0    0         0    0          0       0
12 19:23:30Z  1573673010.3  122    4.4  0.78  0.009   1.0   1.0  100-100  101-101       0      0        0    0         0    0          0       0
13 19:28:30Z  1573673310.3  119    3.0  0.71  0.009   1.0   1.0  100-100  101-101       0      0        0    0         0    0          0       0
14 19:33:30Z  1573673610.2  114    1.7  0.65  0.009   1.0   1.0  100-100  101-101       0      0        0    0         0    0          0       0
15 19:38:40Z  1573673920.6  116    0.3  0.58  0.008   1.0   1.0  100-100  101-101       0      0        0    0         0    0       0.13       0
16 19:43:30Z  1573674210.4  117      0  0.53  0.008   1.0   1.0  100-100  101-101       0      0        0    0         0    0       0.03    0.01
17 19:48:30Z  1573674510.4  118      0  0.48  0.008   1.0   1.0  100-100  101-101       0      0        0    0         0    0          0       0
18 19:53:30Z  1573674810.3  116      0  0.44  0.007   1.0   1.0  100-100  101-101       0      0        0    0         0    0          0       0
19 19:58:30Z  1573675110.4  116      0  0.38  0.007   1.0   1.0  100-100  101-101       0      0        0    0         0    0          0       0
20 20:03:30Z  1573675410.6  117      0  0.35  0.006   1.0   1.0  100-100  101-101    0.06   0.02        0    0         0    0       0.37    0.29
21 20:08:30Z  1573675710.4  122      0  0.33  0.006   1.0   1.0  100-100  101-101     0.4   0.36        0    0         0    0       1.05    0.97
22 20:13:30Z  1573676010.3  124      0  0.36  0.006   1.0   1.0  100-100  101-101    0.32    0.3        0    0         0    0       1.05    1.05
23 20:18:30Z  1573676310.4  125      0   0.4  0.005   1.0   1.0  100-100  101-101    0.34   0.32        0    0         0    0       1.05    1.05
24 20:23:30Z  1573676610.3  125      0  0.44  0.005   1.0   1.0  100-100  101-101    0.16   0.14        0    0         0    0       1.05    1.05
25 20:28:30Z  1573676910.8  125      0  0.48  0.005   1.0   1.0  100-100  101-101    0.14    0.1        0    0         0    0       1.05    1.05
26 20:33:30Z  1573677210.3  124      0  0.53  0.005   1.0   1.0  100-100  101-101       0      0        0    0         0    0          0    0.19
27 20:38:30Z  1573677510.4  123      0   0.5  0.005   1.0   1.0  100-100  101-101       0      0        0    0         0    0          0    0.19
28 20:43:30Z  1573677810.3  125      0  0.48  0.005   1.0   1.0  100-100  101-101    0.16   0.14        0    0         0    0       0.57    0.53
29 20:48:30Z  1573678110.7  127      0  0.48  0.005   1.0   1.0  100-100  101-101    0.28   0.26        0    0         0    0       0.81    0.77
30 20:53:30Z  1573678410.3  121      0  0.46  0.005   1.0   1.0  100-100  101-101       0      0        0    0         0    0          0       0
31 20:58:30Z  1573678710.3  110      0  0.43  0.005   1.0   1.0  100-100  101-101       0      0        0    0         0    0          0       0
32 21:03:30Z  1573679010.7  111      0  0.41  0.005   1.0   1.0   80- 80  101-101    0.08      0        0    0         0    0       0.53       0
-------------------------------------------------------------------------------------------------------------------------------------------------
Totals:                                                                                                          0.6  0.0       0.76    0.91
```

This table lists the main comparisons between the original case and the emulated case. In the target columns the last row shows that the original target was reduced from 100 to 80 because sport was over. As the variant continued with 101 just ignore that last time-step in the final judgement. The Insulin required is slightly lower as expected due to a slightly higher target. The SMB columns confirm that all of the original SMBs were disabled. The tempBasal now has to offset that lack of insulin around those time-steps and is otherwise slightly reduced because of reduced requirements. The totals row at the bottom gives an indication of the changes but the timing of the major differences is important. While the SMBs of 0.6U were delivered the basals added up to 0.12U in those 15 minutes giving a total of 0.72U. In the variant case the total basal insulin adds up to 0.31U in that short period. So in summary this difference is too small to justify a change in settings which worked well on that day and the next week will be different anyway because the body will be in a different condition. But the process could be repeated for the logfiles of the equivalent activities on other days.

The graphical representation if the comparison tells the same story. The best evaluation method is to look at both, the graph and the table when assessing the comparison. The graph for the example looks like this:

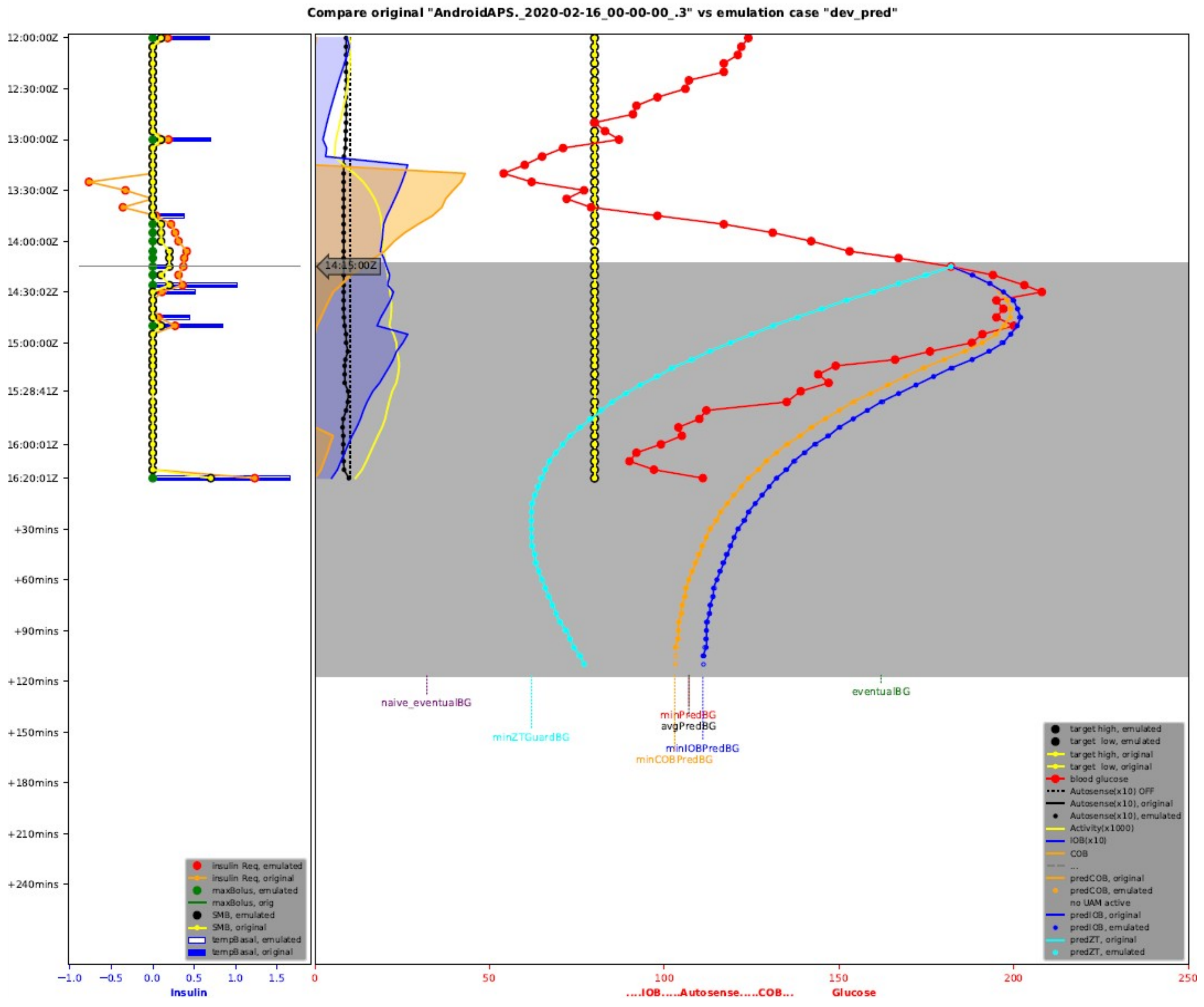
Compare original "AndroidAPS._2019-11-13_00-00-00_.7" vs emulation case "Demo_Sports_Adaptations"



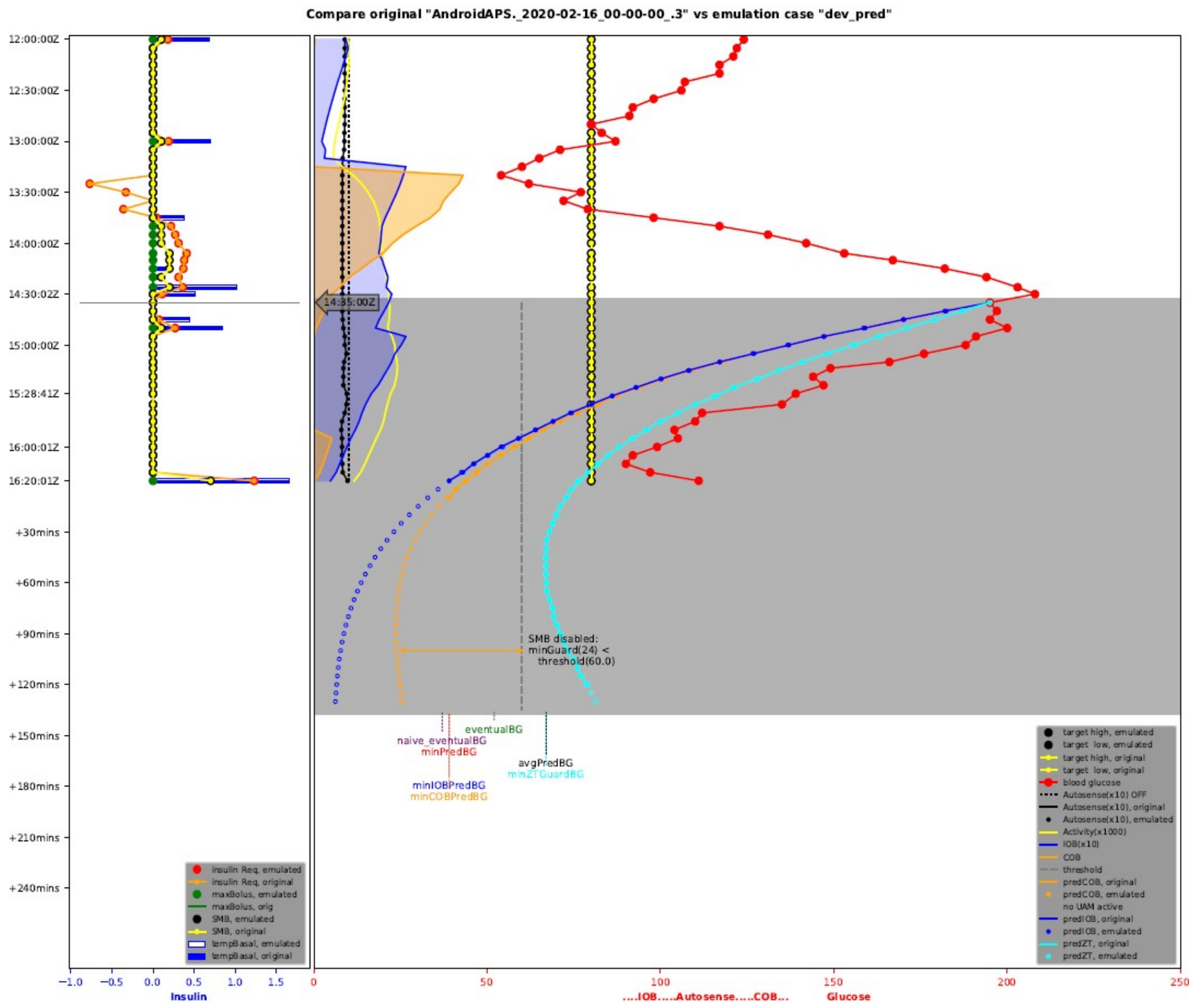
The „pred“ output option

It overlays the graph with the prediction lines known from the AAPS “Home“ tab. With the backward and forward arrows in the pdf-viewer you can get a slow motion animation of the history. The hope is that this will improve the understanding of how the loop works.

The example below is from a different logfile with no variation active and shows a moment when the predictions matched the future glucose quite well. This means the settings matched real life quite well for about one hour and that there was no outside disturbance like new carbs or target changes.



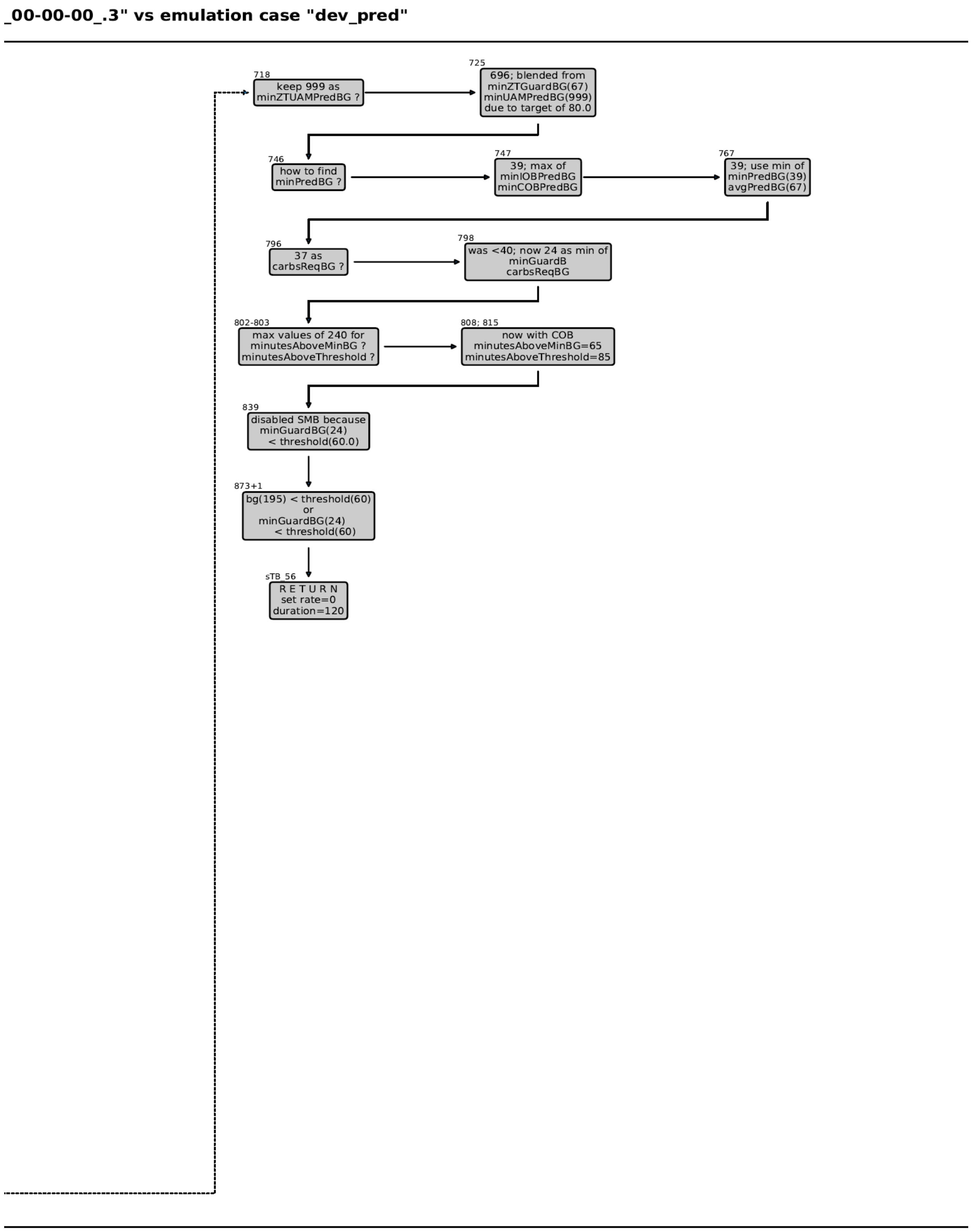
Some time-steps later the loop reports in its „SMB“ tab a reason why SMB was disabled. Such a situation is included in the graph whenever it applies to help understanding of the reasoning. What I find interesting is that the loop compares the predictions against the threshold rather than the lower target. That threshold is defined as 50% of the lower target plus 20mg/dl, i.e. always below even the lower target!



By the way, open circles in the predictions show the initial calculations which later get capped, truncated and rounded in AAPS and those final values are shown as filled circles.

The „flowchart“ output option

In order to better understand the logic and decisions in determine-basal I added this extra graphic. As for the „pred“ option a new flowchart is created at each time step. As an example here is a zoomed in region for the same time-step as before, i.e. the disabling of SMB at 14:35:00Z;


_00-00-00_.3" vs emulation case "dev_pred"
718
keep 999 as minZTUAMPredBG ?
725
696; blended from minZTGuardBG(67) minUAMPredBG(999) due to target of 80.0
746
how to find minPredBG ?
747
39; max of minIOBPredBG minCOBPredBG
767
39; use min of minPredBG(39) avgPredBG(67)
796
37 as carbsReqBG ?
798
was <40; now 24 as min of minGuardB carbsReqBG
802-803
max values of 240 for minutesAboveMinBG ? minutesAboveThreshold ?
808; 815
now with COB minutesAboveMinBG=65 minutesAboveThreshold=85
839
disabled SMB because minGuardBG(24) < threshold(60.0)
873+1
bg(195) < threshold(60) or minGuardBG(24) < threshold(60)
sTB_56
R E T U R N set rate=0 duration=120

Flowchart and decision logic at time 14:35:00Z

It shows the source of the decision in row 839[3] of the original determine-basal.js.

3 This row numbering was valid in AAPS version 2.6 and not updated from version 2.7 onwards

Execute the analysis using the GUI front end

Start by clicking on the shortcut created on the desktop during the installation process. This opens a DOS window. The DOS window should remain empty but may be error messages will show up and I recommend to check for those before quitting the GUI. I tried to catch such messages in the code for display in the form itself but you never know.

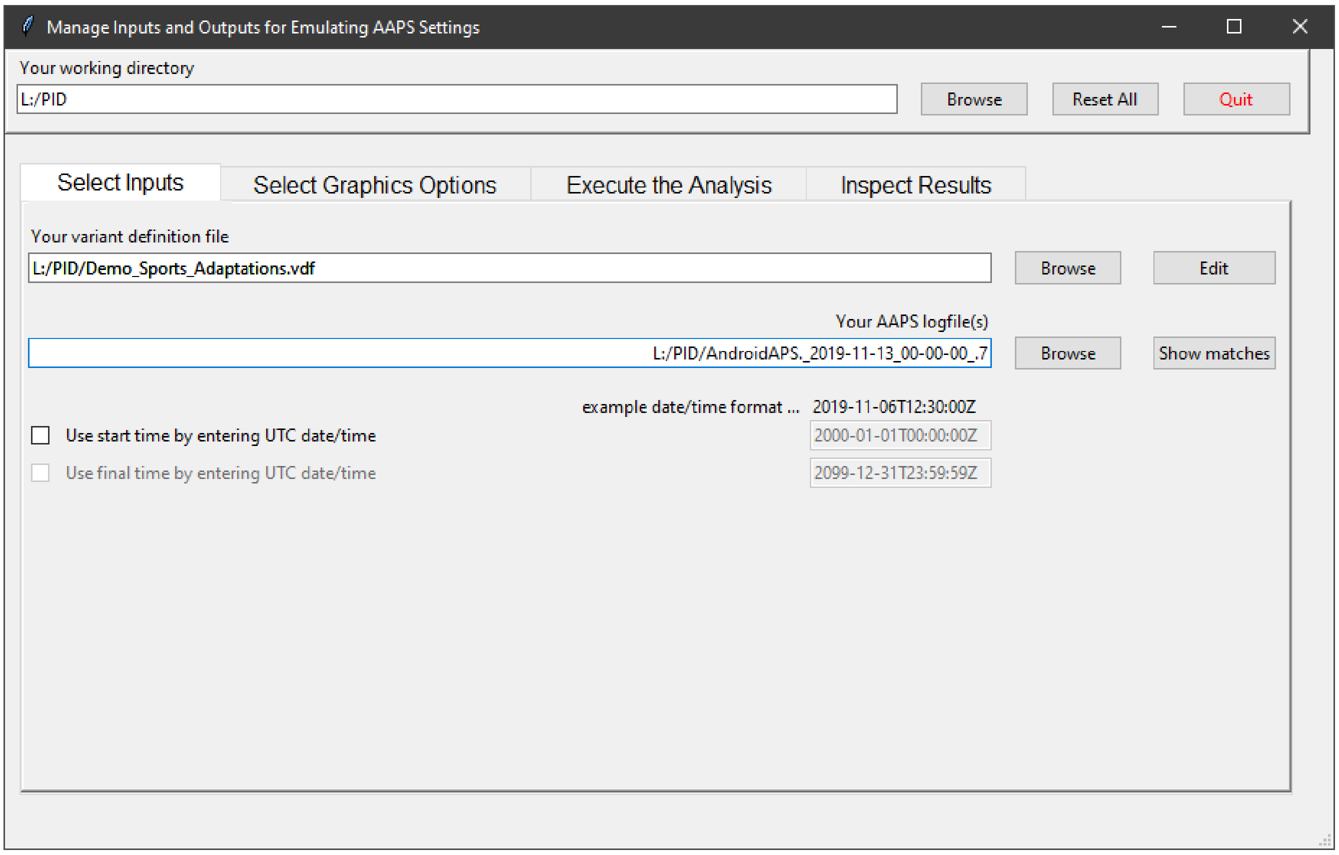
After opening that DOS window another window opens which contains a form itself to interface the whole process. The size of the window may be adjusted so longer filenames are displayed without scrolling. In general, boxes with white background are used to enter text, mostly filenames. Boxes with rounded corners and grey background are command buttons to start certain actions:

- Browse - will start the standard file selection dialogue to find the respective file
- Edit - loads that file into your standard Editor (as long as your system has assigned a standard app for that file type)
- Show - similar to „Edit“ apart from the result PDF where it uses your standard PDF viewer

The process goes from top to bottom and from left to right but of course you can jump to wherever. First, you select the working folder, i.e. the one containing your the variant definition file and later the result files.

Select inputs

Click the „Select Inputs“ tab if not already active. The two required inputs are the names of the variant definition file and the AAPS logfile(s).


Manage Inputs and Outputs for Emulating AAPS Settings

Your working directory: L:/PID

Your variant definition file: L:/PID/Demo_Sports_Adaptations.vdf

Your AAPS logfile(s): L:/PID/AndroidAPS._2019-11-13_00-00-00_.7

example date/time format ... 2019-11-06T12:30:00Z

Use start time by entering UTC date/time: 2000-01-01T00:00:00Z

Use final time by entering UTC date/time: 2099-12-31T23:59:59Z

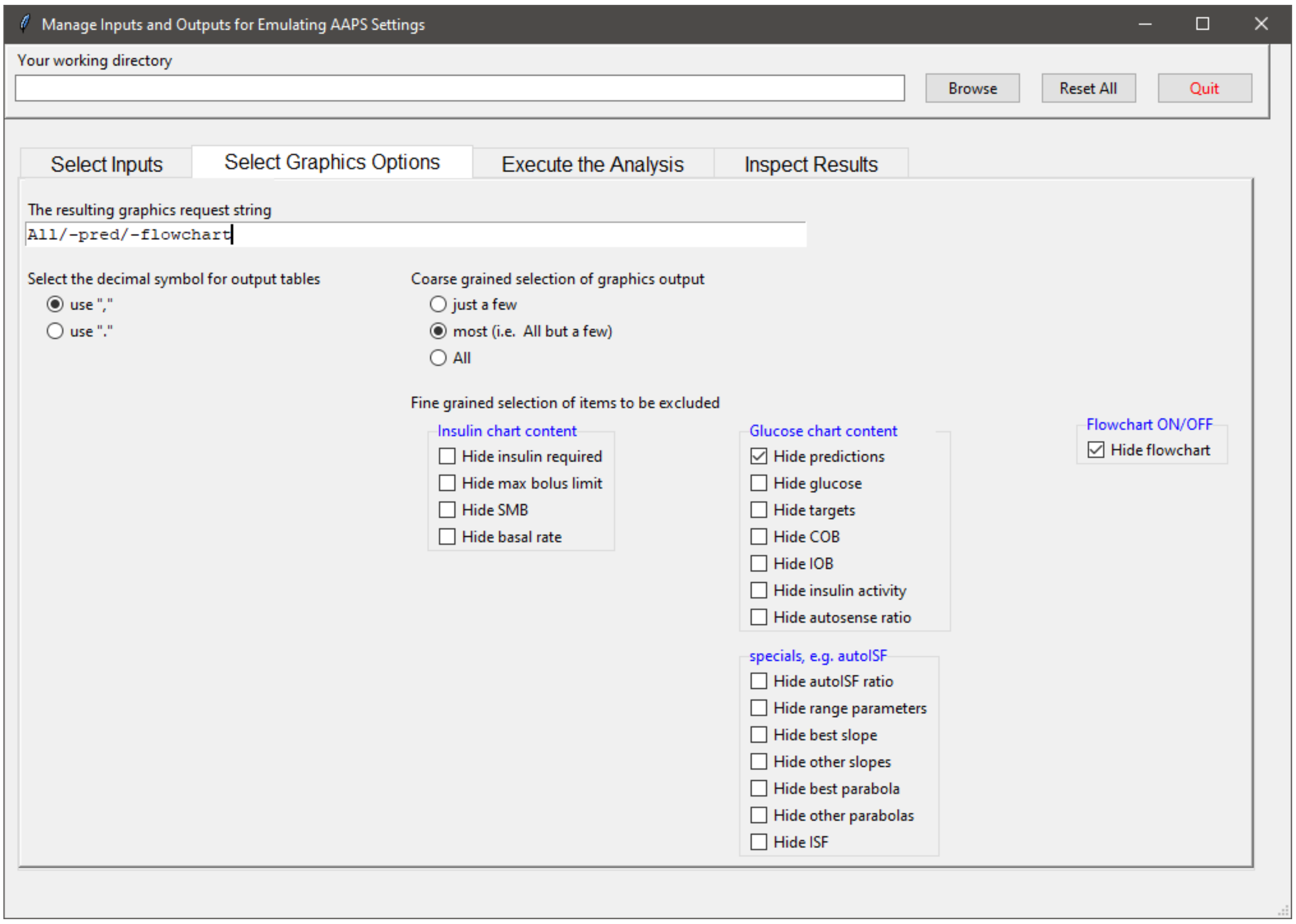
If you need to scan several logfiles for longer time windows you edit the filename by using „*“ or „?“ in the suitable position of the filename. In such cases you can check which files will match the wild card specification by clicking the button „Show matches“.

For specifying an optional start date/time you activate the tick box. The time field will no longer be greyed out and you can enter/edit the start date and time. Above the field you see the format of how to specify the date and time. Remember AAPS uses the internal UTC format which is Greenwich time without daylight savings offset., e.g. in MESZ (Middle-European-SummerTime) it is numerically 2 hours behind. Normally I use copy and paste from that part of the logfile field and then do the detailed adjustments. If final time time is also wanted, again tick the box, copy and paste from the start time and adapt it.

Clicking „Reset All" will clear all input fields and return to the defaults.

Select graphics options

Click the „Select Graphics Options" tab if not already active. This form selects what information to show in the graphic output and a table on screen. It uses radio buttons in the coarse selection groups and where appropriate shows tick boxes for details to be included or excluded. The resulting string as used in the DOS method is dynamically updated to reflect the tick actions but could also be edited manually.



The resulting graphics request string: `All/-pred/-flowchart`

Select the decimal symbol for output tables: [✓ use ","] use "."

Coarse grained selection of graphics output: just a few [✓ most (i.e. All but a few)] All

Fine grained selection of items to be excluded

Insulin chart content
- ☐ Hide insulin required
- ☐ Hide max bolus limit
- ☐ Hide SMB
- ☐ Hide basal rate

Glucose chart content
- ☑ Hide predictions
- ☐ Hide glucose
- ☐ Hide targets
- ☐ Hide COB
- ☐ Hide IOB
- ☐ Hide insulin activity
- ☐ Hide autosense ratio

specials, e.g. autoISF
- ☐ Hide autoISF ratio
- ☐ Hide range parameters
- ☐ Hide best slope
- ☐ Hide other slopes
- ☐ Hide best parabola
- ☐ Hide other parabolas
- ☐ Hide ISF

Flowchart ON/OFF
- ☑ Hide flowchart

The radio buttons for the decimal symbol allow you to use your familiar view and to select the separator used in your favourite spreadsheet when creating the CSV-file.

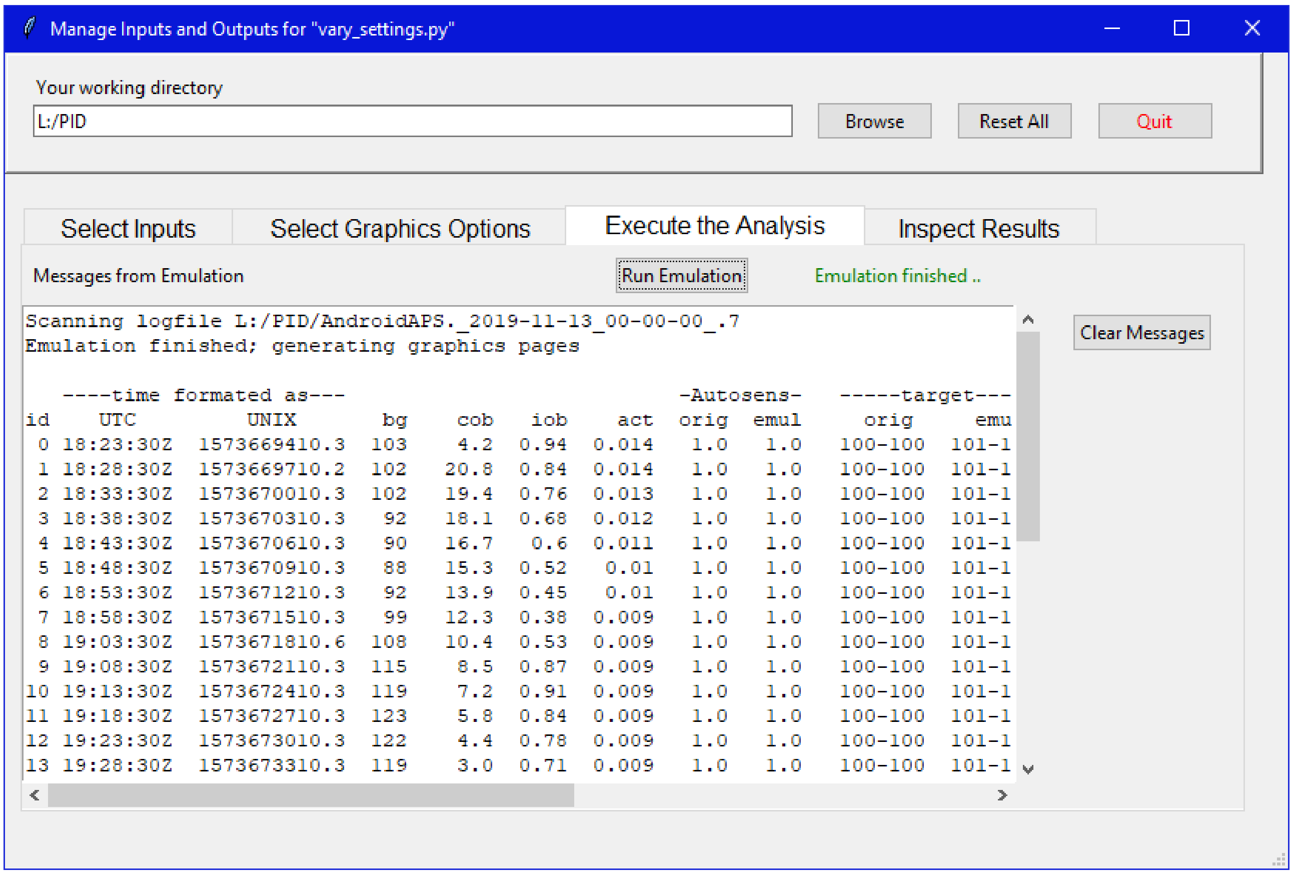
The initial selection for desired outputs is „most" which is the most frequent option. In mode „most" you tick those options you are not interested in.

In mode „just a few" it is the opposite, i.e. you select those options you are specifically interested in.

In mode „All" the fine grained selections in the bottom are disabled and hidden because they make no sense in that context.

Execute the analysis

Click the „Execute the Analysis" tab if not already active. The button „Run Emulation" will kick off the analysis. The large white box shows messages otherwise displayed in the DOS window. Here it may make sense to adjust the window size to show more content without scrolling. The vertical resizing is limited to 30 lines so the window fits on a normal monitor.



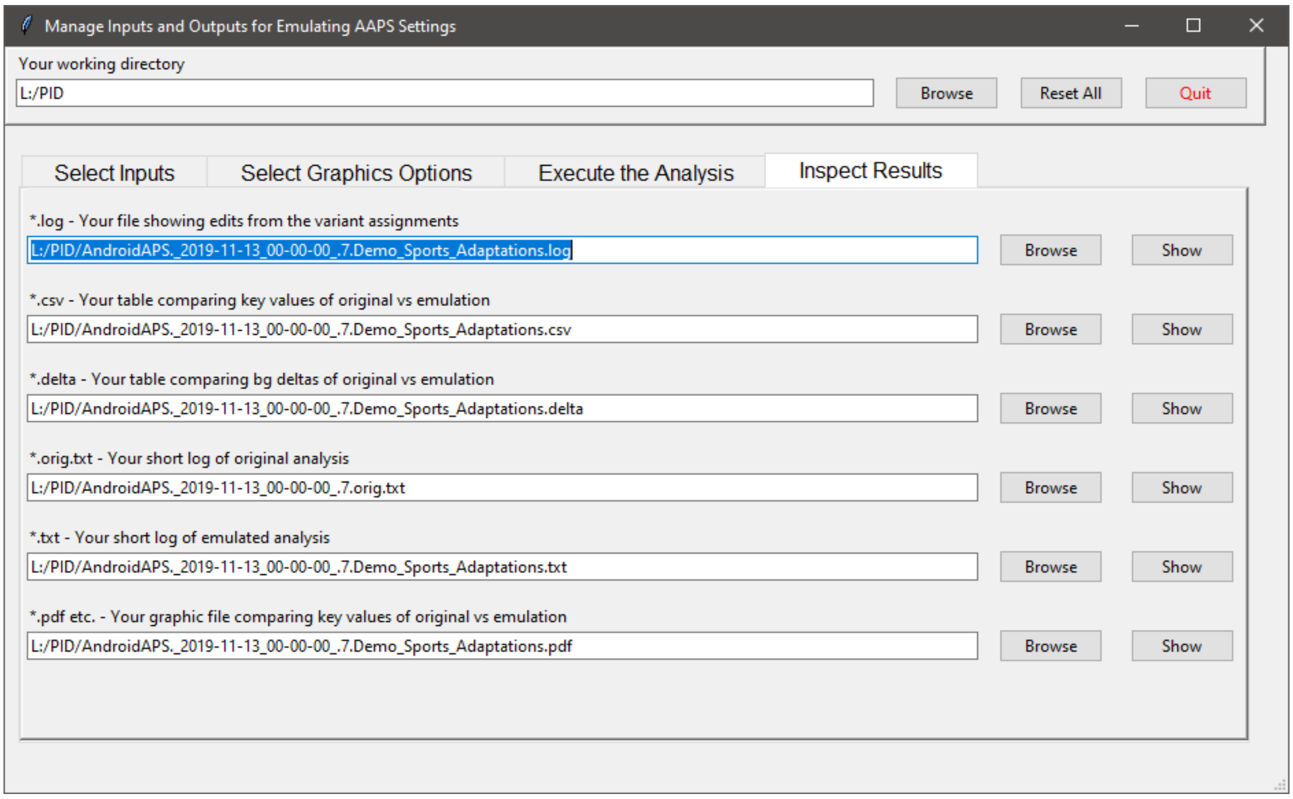
```
Scanning logfile L:/PID/AndroidAPS._2019-11-13_00-00-00_.7
Emulation finished; generating graphics pages

   ----time formated as---                         -Autosens-   -----target---
id     UTC          UNIX       bg     cob   iob     act  orig   emul     orig     emu
 0 18:23:30Z  1573669410.3  103    4.2  0.94  0.014   1.0    1.0   100-100  101-1
 1 18:28:30Z  1573669710.2  102   20.8  0.84  0.014   1.0    1.0   100-100  101-1
 2 18:33:30Z  1573670010.3  102   19.4  0.76  0.013   1.0    1.0   100-100  101-1
 3 18:38:30Z  1573670310.3   92   18.1  0.68  0.012   1.0    1.0   100-100  101-1
 4 18:43:30Z  1573670610.3   90   16.7   0.6  0.011   1.0    1.0   100-100  101-1
 5 18:48:30Z  1573670910.3   88   15.3  0.52   0.01   1.0    1.0   100-100  101-1
 6 18:53:30Z  1573671210.3   92   13.9  0.45   0.01   1.0    1.0   100-100  101-1
 7 18:58:30Z  1573671510.3   99   12.3  0.38  0.009   1.0    1.0   100-100  101-1
 8 19:03:30Z  1573671810.6  108   10.4  0.53  0.009   1.0    1.0   100-100  101-1
 9 19:08:30Z  1573672110.3  115    8.5  0.87  0.009   1.0    1.0   100-100  101-1
10 19:13:30Z  1573672410.3  119    7.2  0.91  0.009   1.0    1.0   100-100  101-1
11 19:18:30Z  1573672710.3  123    5.8  0.84  0.009   1.0    1.0   100-100  101-1
12 19:23:30Z  1573673010.3  122    4.4  0.78  0.009   1.0    1.0   100-100  101-1
13 19:28:30Z  1573673310.3  119    3.0  0.71  0.009   1.0    1.0   100-100  101-1
```

This tabular listing looks slightly different with the later software versions.

Error messages or hints about missing inputs are displayed in red. The button „Clear Messages" obviously will clear all contents in the message box. This is useful before starting the next analysis.

Inspect results

Click the „Inspect Results“ tab if not already active. This tab will also be activated automatically once the analysis is finished. Also the filenames are already prefilled after the analysis. However, you may use the „Browse“ buttons to select and see content of analyses run before.


Manage Inputs and Outputs for Emulating AAPS Settings

Your working directory
L:/PID

*.log - Your file showing edits from the variant assignments
L:/PID/AndroidAPS._2019-11-13_00-00-00_.7.Demo_Sports_Adaptations.log

*.csv - Your table comparing key values of original vs emulation
L:/PID/AndroidAPS._2019-11-13_00-00-00_.7.Demo_Sports_Adaptations.csv

*.delta - Your table comparing bg deltas of original vs emulation
L:/PID/AndroidAPS._2019-11-13_00-00-00_.7.Demo_Sports_Adaptations.delta

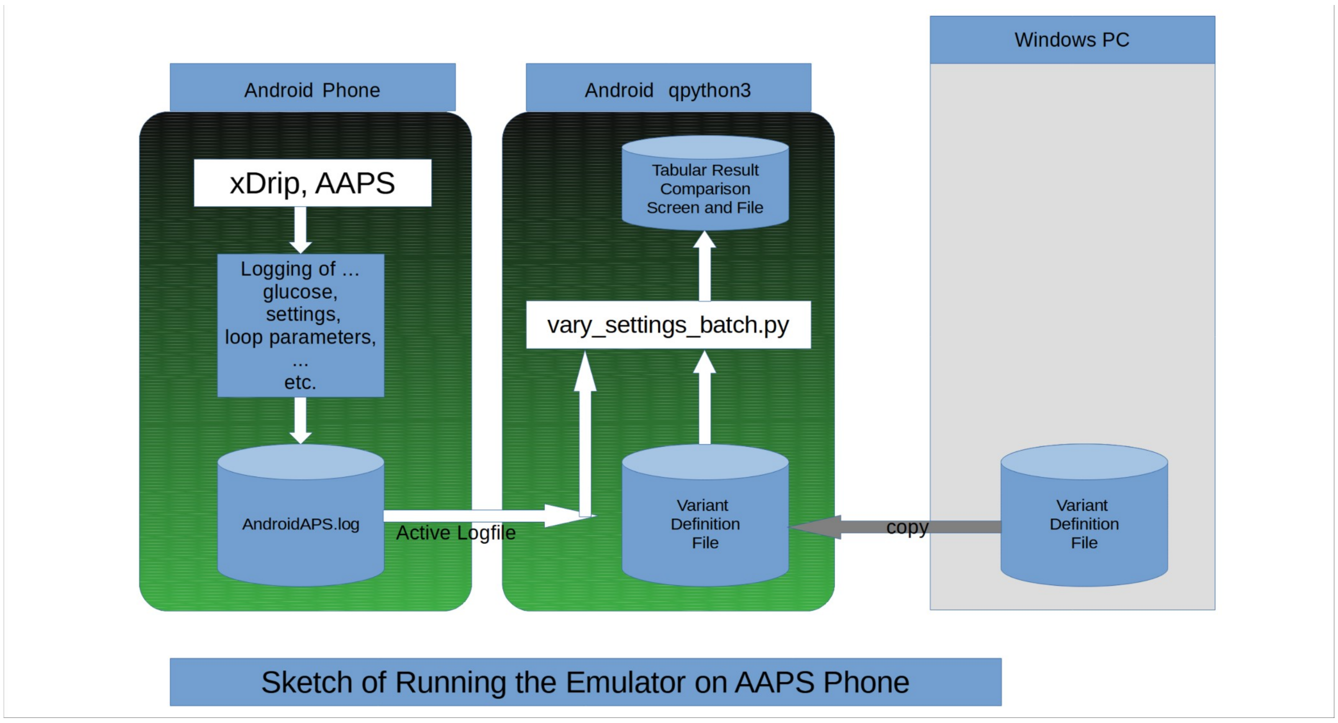
*.orig.txt - Your short log of original analysis
L:/PID/AndroidAPS._2019-11-13_00-00-00_.7.orig.txt

*.txt - Your short log of emulated analysis
L:/PID/AndroidAPS._2019-11-13_00-00-00_.7.Demo_Sports_Adaptations.txt

*.pdf etc. - Your graphic file comparing key values of original vs emulation
L:/PID/AndroidAPS._2019-11-13_00-00-00_.7.Demo_Sports_Adaptations.pdf

Executing the analysis on the AAPS phone

Here is a sketch of the data and process flows when using the emulator on the Android phone running AAPS:

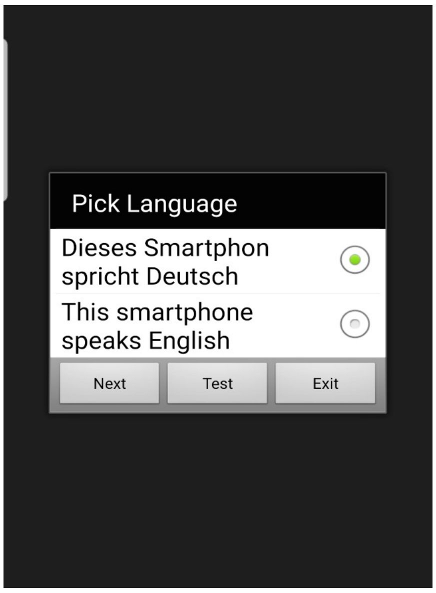


Using the emulator on the phone is much easier than on Windows because only very few input parameters are required. These few are input via initial dialogues. Also no logfile input is needed because it always works with the currently active logfile. This way the emulator is effectively shadowing the active loop and you can see immediately whether the alternative settings would have delivered different SMBs or TBRs.

If SMB values are higher or lower than in the master the user will be informed by a message from the speech synthesis. Thus you can carry the phone hidden away until alarmed and if so judge whether you want to apply the incremental SMB as an extra bolus manually.

As a further safety measure the logfile is scanned regularly for those „add'l carbs req“. If found for the last loop execution then again a message is output via the speech synthesis.

Start the emulation



Pick Language
Dieses Smartphon spricht Deutsch
This smartphone speaks English
Next
Test
Exit

On the phone press the „QPython3L“ button created during installation. There, press Programs , select „vary_settings_batch.py“ and finally select „Run“.

The first dialogue is used to select the language for the speech synthesis. Why that? Initially I set it to German but sometimes the lady insisted on pronouncing everything in English. Occasionally she even switched in the middle of a session.

That meant I had to restart the script. I suspect those switches were triggered by xDrip+ when „Speak Readings“ was active.

Click „Test“ to listen to a sample speech synthesis.

Click „Next“ to proceed.

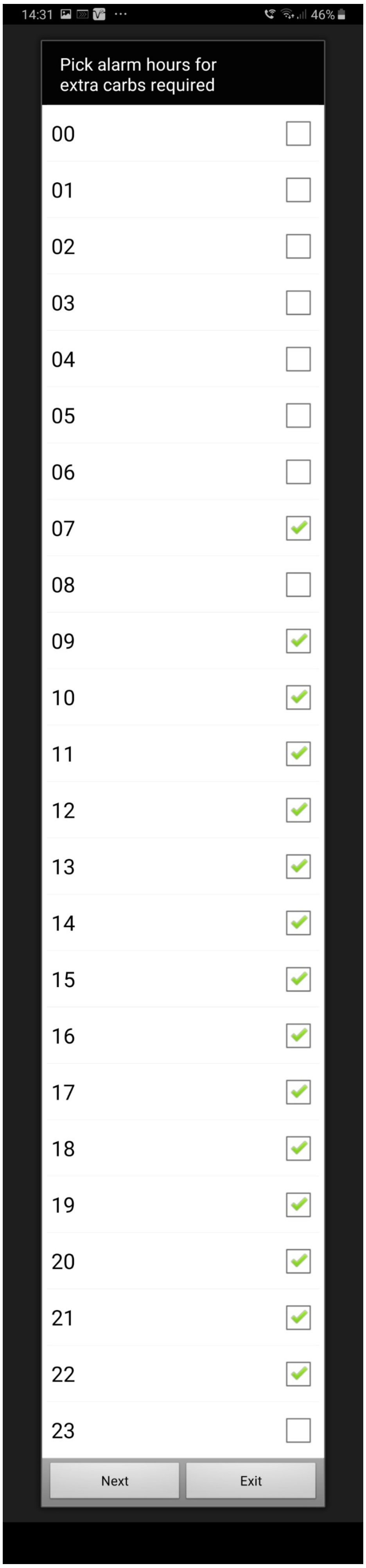
The next three dialogues serve to select all those hours of the day during which you want to allow spoken alarms. For example you can disable alarms during sleep times.

There are three situations which trigger spoken alarms:

- Announce „add'l carbs required“ if discovered during the scan of the AAPS messages
- Announce an incremental bolus amount if the SMB would have been higher
- Announce a reduced bolus (sorry – too late now) if the SMB would have been less

The screen copy to the right shows the example for the first case, the extra carbs required.

Obviously you need to scroll up and down to reach all items on that list.


14:31 46%

Pick alarm hours for extra carbs required

00	☐
01	☐
02	☐
03	☐
04	☐
05	☐
06	☐
07	✓
08	☐
09	✓
10	✓
11	✓
12	✓
13	✓
14	✓
15	✓
16	✓
17	✓
18	✓
19	✓
20	✓
21	✓
22	✓
23	☐

The next dialogue window which opens is for picking all those items you want listed on the phones screen. You can see the initial default selection. Depending on the size of your display in landscape orientation you may not be able to fit everything into a single row per loop and may accordingly deselect less interesting columns.

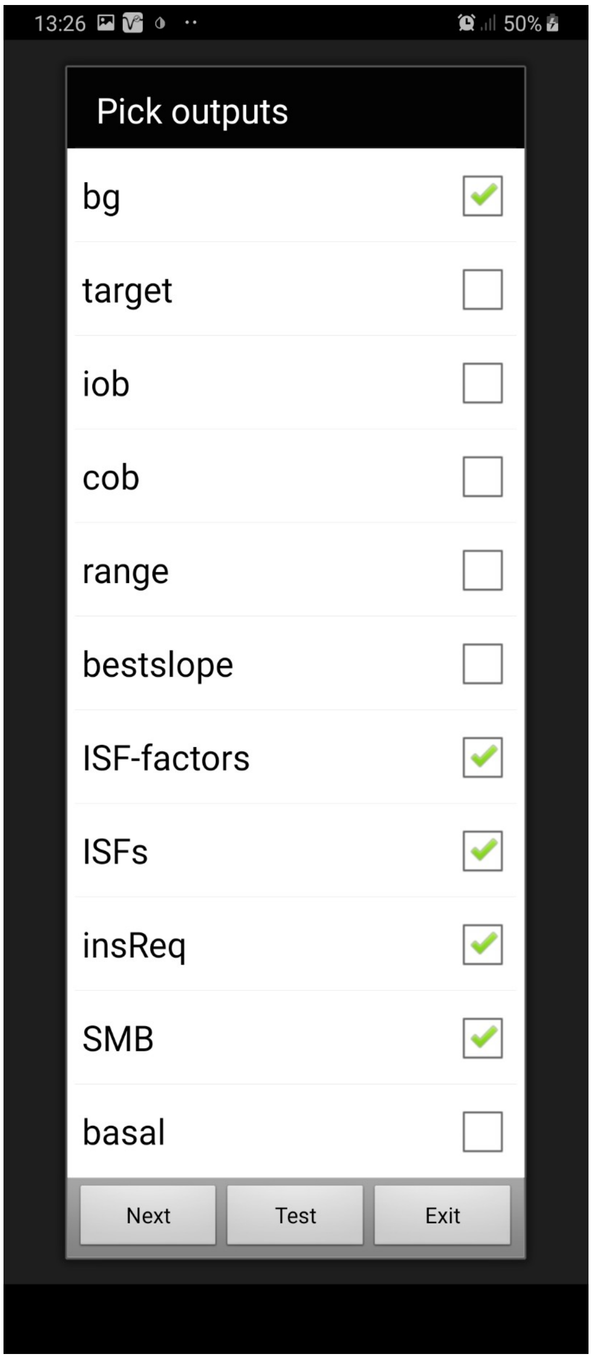
You may have to scroll the screen up and down to reach all the options.

The initial default selection is suited to the autoISF[4] setup and you most likely change those picks.

This default set requires 93 columns in the display.

Click „Test“ to generate a speech output announcing the required number of columns for the picks chosen.

Click „Next“ to proceed.

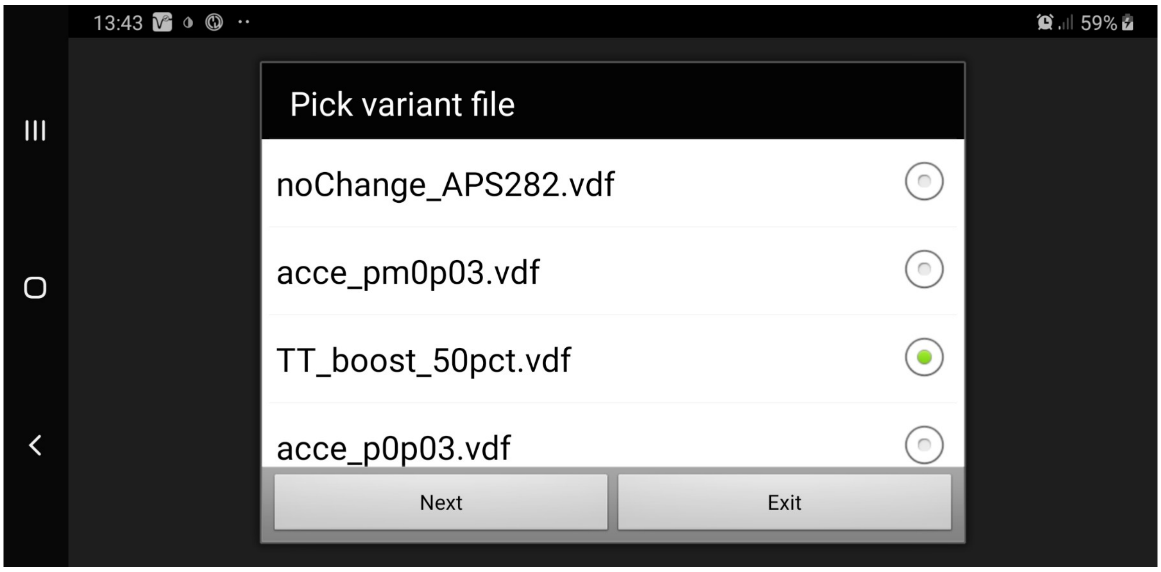

13:26 50%

Pick outputs

bg [✓]
target [ ]
iob [ ]
cob [ ]
range [ ]
bestslope [ ]
ISF-factors [✓]
ISFs [✓]
insReq [✓]
SMB [✓]
basal [ ]

4 See my branch autoISF

The last dialogue window which opens is for picking your single variant definition file. This is a convenient time to rotate the phone to landscape to see longer filenames nicely displayed and to prepare for the result table display.

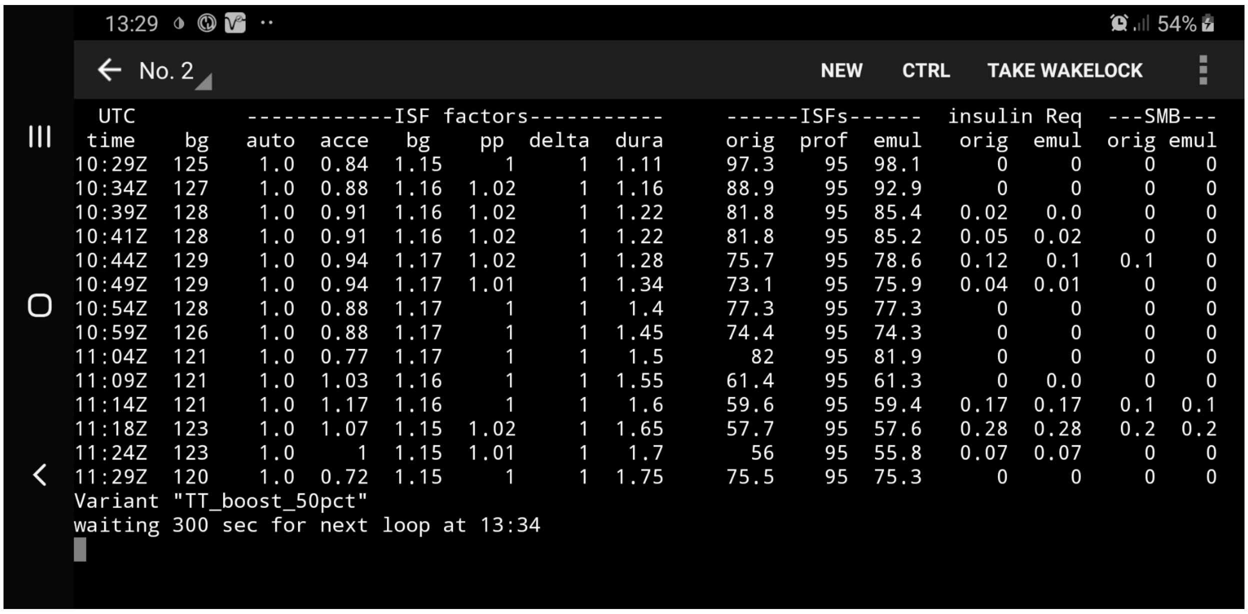


Click „Next“ to proceed.

The results

The regular result files are saved in the logfile folder in case you need them. Please note that no graphics file is created as I could not find the matplotlib library for Android.

The main result of the Android version is the life table with key results. They cover the last hour or so. Older entries get deleted and newer ones get appended. This table is now also printed on Windows although not limited to max 14 rows.



```
  UTC         ------------ISF factors-----------      ------ISFs------  insulin Req ---SMB---
 time    bg  auto  acce    bg    pp delta  dura    orig  prof  emul  orig  emul orig emul
10:29Z  125   1.0  0.84  1.15     1     1  1.11    97.3    95  98.1     0     0    0    0
10:34Z  127   1.0  0.88  1.16  1.02     1  1.16    88.9    95  92.9     0     0    0    0
10:39Z  128   1.0  0.91  1.16  1.02     1  1.22    81.8    95  85.4  0.02   0.0    0    0
10:41Z  128   1.0  0.91  1.16  1.02     1  1.22    81.8    95  85.2  0.05  0.02    0    0
10:44Z  129   1.0  0.94  1.17  1.02     1  1.28    75.7    95  78.6  0.12   0.1  0.1    0
10:49Z  129   1.0  0.94  1.17  1.01     1  1.34    73.1    95  75.9  0.04  0.01    0    0
10:54Z  128   1.0  0.88  1.17     1     1   1.4    77.3    95  77.3     0     0    0    0
10:59Z  126   1.0  0.88  1.17     1     1  1.45    74.4    95  74.3     0     0    0    0
11:04Z  121   1.0  0.77  1.17     1     1   1.5      82    95  81.9     0     0    0    0
11:09Z  121   1.0  1.03  1.16     1     1  1.55    61.4    95  61.3     0   0.0    0    0
11:14Z  121   1.0  1.17  1.16     1     1   1.6    59.6    95  59.4  0.17  0.17  0.1  0.1
11:18Z  123   1.0  1.07  1.15  1.02     1  1.65    57.7    95  57.6  0.28  0.28  0.2  0.2
11:24Z  123   1.0     1  1.15  1.01     1   1.7      56    95  55.8  0.07  0.07    0    0
11:29Z  120   1.0  0.72  1.15     1     1  1.75    75.5    95  75.3     0     0    0    0
Variant "TT_boost_50pct"
waiting 300 sec for next loop at 13:34
```

Sometimes when rotating the phone the column alignment was lost and I had to wait for the next loop update. The reason may be that on my phone the maximum with in portrait mode is 46, i.e. exactly half the above table width. Therefore I append a BLANK and rotating the phone is less annoying. If you have different screen dimensions and are annoyed let me know and I may consider adding a user specific config file.

Stop the emulation

Click the “Back“-button of the phone and then select „NO“, i.e. do not run it in the background.

Tips and tricks

- If the keyboard overlays or pushes the result table off screen click the phones „Back“button
- If you rotate the phone between portrait and landscape orientation the column alignment may get screwed. This will be fixed upon the next loop execution and look nice again.
- If initially the table contains only a few lines then the script was started shortly after a new logfile was started. This fresh logfile covers less than 1 hour.
- The first column lists the loop execution times in UTC format. So for daylight savings time in central Europe this label is 2 hours behind your local time and that of the phone.
- Sometimes the qpython3-app disappears from the list of recent apps (on my phone there seems to be a limit of 4). In such cases I killed the app (Android Settings → Apps → Qpython3 → Force Stop) and restarted it.
- The message „add'l carbs req“ appears even although sufficient carbs were input in the bolus calculator. I included a filter to suppress it for smaller amounts and longer time spans. If the message still is output via the speech synthesis I check whether I really need fast carbs on top or whether the carbs on board are fast enough to keep me safe.

Status: 30-Jun-2022 @ 14:40

Appendix A – How to use logfiles created without autoISF or versions before 2.2.6

Under his heading there are 3 possible scenarios:

Scenario 1: the logfile was created by regular AAPS, i.e. without any autoISF

```
profile         enable_autoISF              False                   ### for logs before autoISF
profile         smb_delivery_ratio          0.5                     ### AAPS default
```

This case was already touched in the sports adaptation example above. The VDF-file needs these two extra lines inserted at the beginning to tell it that autoISF is disabled and that that SMB delivery ratio is as in regular AAPS.

Scenario 2: the logfile was created by AAPS, including an earlier version of autoISF

```
profile        enable_autoISF          profile['use_autoisf']             ### get from previous name
profile        autoISF_max             profile['autoisf_max']             ### get from previous name
profile        dura_ISF_weight         profile['autoisf_hourlychange']    ### get from previous name
profile        enable_dura_ISF_with_COB profile['enableautoisf_with_COB'] ### get from previous name
profile        autoISF_min             -999                               ### no limit at the time
profile        enable_pp_ISF_always    False                              ### no limit at the time
profile        pp_ISF_hours            0                                  ### no limit at the time
new_parameter  pp_ISF                  1                                  ### effectively unused
#ew_parameter  delta_ISF               1                                  ### effectively unused
```

In this case you have the autoISF2.1 algorithm included in AAPS but the logic has changed and the parameter names as well. You need to tell the VDF-file which old name to assign to the corresponding new one. The details really depend on which „interim“ version of autoISF was active at the time. If it does not work read the error message which will tell you which parameter is missing or misspelled.

Scenario 3: the logfile was created by regular AAPS, i.e. without any autoISF, but you want to explore how autoISF would change things

```
profile         enable_autoISF              True                    ### activate all of autoISF
profile         autoISF_min                 0.7
profile         autoISF_max                 1.2
profile         smb_delivery_ratio          0.5
profile         enable_dura_ISF_with_COB    True
profile         dura_ISF_weight             0.5

new_parameter   acce_ISF                    1.0                     ### inactive
new_parameter   bg_ISF                      1.0                     ### inactive
new_parameter   pp_ISF                      1.0                     ### inactive
new_parameter   delta_ISF                   1.0                     ### inactive
```

Here is an example how you would activate the dura_ISF effect and keep the other ones inactive in order to see whether and when it would have changed your insulin delivery.

If alternatively you want to explore how pp_ISF could have helped with your gastroparesis then you would define dura_ISF = 1.0 and instead of the two profile parameters for dura_ISF you list the parameters contributing to pp_ISF.

Appendix P – Define the pump profile

You may want to return to a regular profile if you used automation rules for changing profiles. Thisis how you can define a pump profile inside a VDF-file:

```
STAIR_ISF            00:00:00Z    45          ### 01h_C(entral)E(uropean)T(ime) or 02h_CEST
STAIR_ISF            01:00:00Z    44          ###
STAIR_ISF            02:00:00Z    42          ###
...
STAIR_ISF            17:00:00Z    36          ### 18h_CET
STAIR_ISF            18:00:00Z    38          ###
...
STAIR_ISF            22:00:00Z    43          ### 23h_CET
STAIR_ISF            23:00:00Z    44          ### 00h_CET
profile      sens                 STAIR_ISF   ###

STAIR_CR             00:00:00Z    8.0         ### 01h_C(entral)E(uropean)T(ime) or 02h_CEST
STAIR_CR             01:00:00Z    7.5         ###
...
STAIR_CR             20:00:00Z    7.5         ###
STAIR_CR             21:00:00Z    8.0         ###
STAIR_CR             22:00:00Z    9.0         ### 23h_CET
STAIR_CR             23:00:00Z    9.0         ### 00h_CET
profile      carb_ratio           STAIR_CR    ###

STAIR_BAS            00:00:00Z    0.41        ### 01h_C(entral)E(uropean)T(ime) or 02h_CEST
STAIR_BAS            01:00:00Z    0.43        ###
STAIR_BAS            02:00:00Z    0.44        ###
STAIR_BAS            03:00:00Z    0.50        ###
...
STAIR_BAS            19:00:00Z    0.75        ### 20h_CET
STAIR_BAS            20:00:00Z    0.75        ###
STAIR_BAS            21:00:00Z    0.60        ###
STAIR_BAS            22:00:00Z    0.45        ### 23h_CET
STAIR_BAS            23:00:00Z    0.43        ### 00h_CET
profile      current_basal        STAIR_BAS   ###
```

Some lines were omitted in that example for better readability. The important thing to pay attention to is the timeshift between UTC on one hand and CET (Central European Time) or CEST(Central European Summer Time), respectively, on the other hand. The lines must be sorted by UTC time. Therefore the first line (winter time) or first two lines (summer time) from your pump definition must be cut off and appended at the end. If your profile ist not fully populated for 24 hours this may also mean you first need to create a pump entry at 01 or 02 hours, respectively.

Appendix V – Some more examples of VDF entries

This may serve as a collection of ideas and may serve as proformas for your own studies.

Array	Element	Value/Expression	Comment
Examples how to change targets			
profile	max_bg	85	### fixed value
profile	min_bg	profile[‘min_bg’]+10	### increase current value by 10mg%
profile	min_bg	profile[‘min_bg’]*1.2	### increase current value by 20%
profile	target_bg	(profile[‘min_bg’] + profile[‘max_bg’]) / 2	### correct instruction but not needed as AAPS will do that internally anyway
Further examples for attributes of profile			
profile	enableSMB_always	True	### would work for all CGMs; note the upper case in True
profile	sens	95	### fixed value
profile	carb_ratio	12.5	### fixed value
profile	enableUAM	True	
profile	enableSMB_with_high_temp target	False	
profile	maxUAMSMBBasalMinutes	180	### just theoretically, not possible in AAPS
Example of changing ISF depending on the BG deviation			
temp	BG	glucose_status[‘glucose’]	### save current glucose as temporary variable BG
temp	Aim	(profile[‘min_bg’] +profile[‘max_bg’])/2	### save average target as temporary variable Aim
temp	Dev	(temp[‘BG’] - temp[‘Aim’])/temp[‘Aim’]	### save relative deviation as temporary variable Dev
profile	sens	profile[‘sens’] / temp[‘Dev’]	### strengthen or weaken ISF as given by the relative deviation
Example of strengthening ISF if delta>10mg/dl/5min and glucose>100mg/dl			
temp	steep	glucose_status[‘delta’] > 10	### 1 if delta is above 10mg/dl/5min ; 0 otherwise
temp	high	glucose_status[‘glucose’] > 100	### 1 if glucose is above 100mg/dl; 0 otherwise
temp	strong	1 + temp[‘high’] * temp[‘steep’] * 0.2	### 1.2 if both conditions are satisfied; 1 otherwise
profile	sens	profile[‘sens’] / temp[’strong’]	### strengthen ISF by 20% if both conditions are satisfied; no change otherwise
Examples of changes depending on time			
STAIR	2020-04-14T03:00:00Z	120	### value starting at 3am UTC, equals 05:00 CEST to prepare for exercise
STAIR	2020-04-14T05:00:00Z	150	### value starting at 07:00 CEST, the start of early morning exercise
STAIR	2020-04-14T06:00:00Z	90	### value starting at 08:00, the end of early morning exercise
profile	max_bg	STAIR	### time dependent value between 5 and 8 o'clock CEST
INTERPOL	2020-04-14T20	0	### value at 20:00 UTC
INTERPOL	2020-04-14T22	4	### value at 22:00 UTC
profile	carb_ratio	INTERPOL+10	### linearly rising CR including interpolation and extrapolation
Further possibilities			
new_parameter	AAPS_Version	"<2.7"	### use AAPS 2.6.1 algorithm instead of 2.7 or 2.8; not the double quotes to signify text entry
autosens_data	ratio	1	### effectively turns Autosens off
autosens_data	ratio	1.3	### just theoretically, does not make much sense