Introduction

AutoISF is meant for the advanced user who has a deep understanding of AAPS and who has tuned the system to achieve a TIR of about 90% or better. If such a user is ambitious and wants to improve further, the methods contained can very well help.

This document describes how and when ISF is adapted automatically and provides short descriptions of the weighting factors to tune it to your individual needs. Detailed guides covering special situations and examples will follow later as separate documents including results of using it in full loop mode, i.e. without carb or insulin entry by the user. Furthermore, it describes alernative methods how to scale, activate or deactivate SMBs. Finally there are methods for handling mild up to intensive activity.

The adaptation of ISF is based on special glucose behaviour and results in an adaptation factor just like the Autosense factor. However, here only ISF is adapted and no other setting. The scenarios analysed typically cover the last 10-30 minutes and therefore autoISF reacts much faster to problems or recoveries. Often Autosense would drive me into hypos because of its delayed reaction even after things had come back to target and I disabled Autosense although mathematically both can coexist.

AutoISF is part of oref1 in OpenAPS SMB and cannot coexist with the recent DynamicISF which therefore is included in its own plug-in anyway as an alternative to OpenAPS AMA and OpenAPS SMB.

Please note that these rapid adaptations of ISF render the use of Autosense and Autotune useless because those would draw conclusions based on constant ISF and therefore false assumptions. If you want to use Autotune then disable autoISF for those periods.

There are 4 different effects in glucose behaviour that autoISF checks and reacts to:

1. **acce_ISF** is a factor derived from acceleration of glucose levels
2. **bg_ISF** is a factor derived from the deviation of glucose from target
3. **delta_ISF** and **pp_ISF** are factors derived from glucose rise delta
4. **dura_ISF** is a factor derived from glucose being stuck at high levels

Finally these factors are compared among each other and against Autosense. Normally the strongest of them will be used with some exceptions as detailed further below. These factors work the same way as the Autosense factor works, i.e. the profile sensitivity ISF is divided by the factor to deliver a final sensitivity ISF.

In the SMB-tab, section Script debug, you can always see which values were assigned to the 4 factors shown above during the last loop execution. It also lists explanations in case the factor had to be modified or why it cannot be used. Some interim values like dura_ISF_average are listed, too. All that can be seen in the SMB-tab, sections Glucose status, Profile and Script debug, respectively.

Again in analogy to Autosense there are upper (*autoISF_ma*x) and lower (*autoISF_min*) limits for how far ISF can be modified in total.

In analogy to enabling SMB there is a setting *enable_autoISF* which determines whether any of the 4 ISF adaptations of autoISF listed above are enabled or none at all.

The settings specific to autoISF are collected in its own menu found at the end of the OpenAPS SMB menu. A screenshot is shown as attachment. Another trick for finding them is to use the filter method at the top of the Preferences page which searches for all settings containing the string you enter in the filter field.

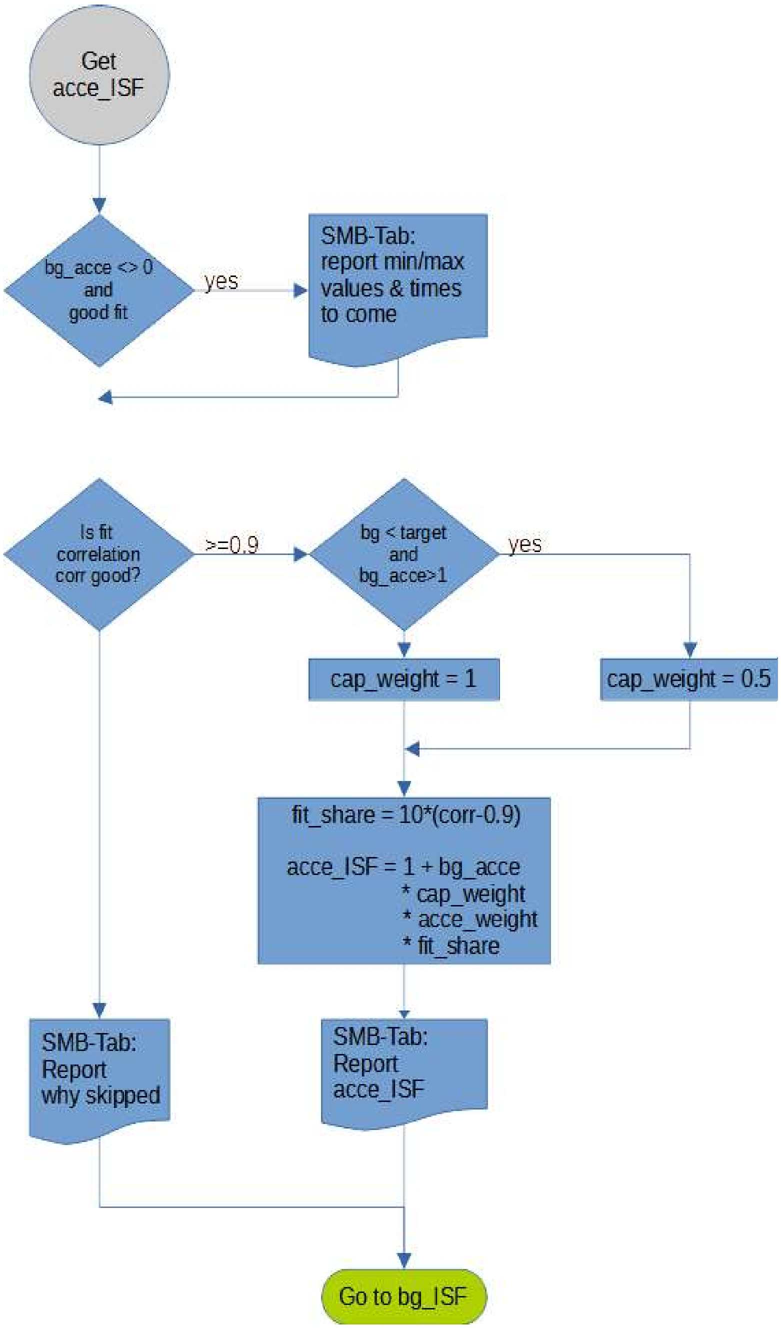
acce_ISF determination and its impact

This is the latest contribution to autoISF and has been in action since late December 2021.

For things like glucose or delta to change, there first must be an acceleration. Therefore acceleration recognises such changes earlier and is used by autoISF to take pre-emptive action.

Glucose and delta, its 1st derivative, play a significant role in AAPS in determining the insulin required. Acceleration, its 2nd derivative, was not included so far. One reason might be that it is harder to extract from the glucose history considering that delta needed to be averaged already to provide a reliable signal. In autoISF a best fit algorithm is used to determine the parabola which best matches the glucose data.

Once the formula for the parabola is known it is then very easy to determine the acceleration. Sometimes the fit has bad correlation, i.e. it deviates too much from the glucose readings. In such cases there is no contribution from acceleration and acce_ISF = 1.



Otherwise acce_ISF is calculated by

acce_ISF = 1 + acce_weight * fit_share * cap_weight * acceleration

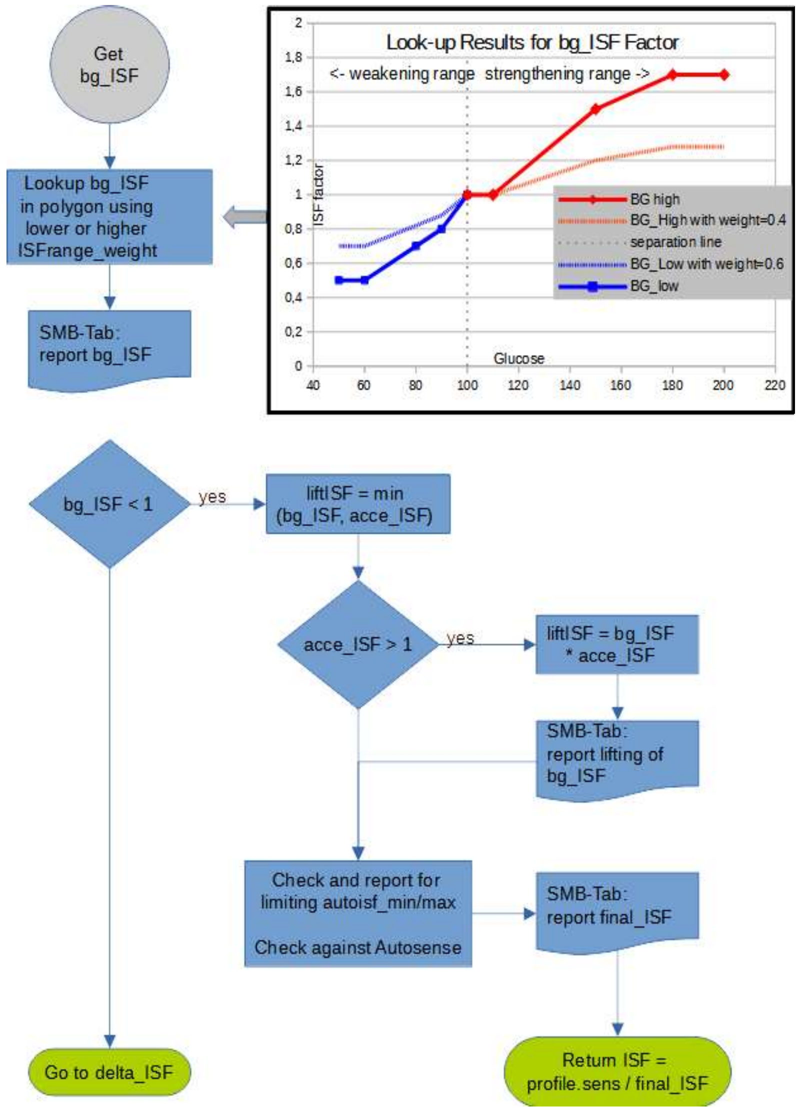
where	fit_share	a measure of fit quality, i.e. 0% if unacceptable up to 100% if perfect;
	cap_weight	is 0.5 below target and 1.0 otherwise;
	acce_weight	is bgAccel_ISF_weight for acceleration away from target, i.e. mostly positive or bgBrake_ISF_weight for acceleration towards target, i.e. mostly negative

Initially I had assumed that the weights for accelerating and braking are of similar size. First experiences say that the weight while decelerating should be 30-40% lower than for acceleration to reduce glucose oscillations. Quite often the acce_ISF contribution plays the dominant role inside autoISF and is therefore very important and delicate. Weights for acce_ISF of 0 disable this contribution. Start small with weights like 0.02 and observe the results before increasing them. Keep in mind that negative acceleration will start to happen while glucose is apparently still rising but the slope reduces. Here, acce_ISF will be <1, i.e. sensitivity grows and less insulin than normal will be required even before the glucose peak is reached.

bg_ISF determination and its impact

There are indicators that higher glucose needs stronger ISF. This was evident from all the successful AAPS users defining automation rules which strengthen the profile at higher glucose levels. The drawback is that there are sudden jumps in ISF at switch points and no further or minor adaptations in between.

In autoISF a polygon is provided that defines a relationship between glucose and ISF and interpolates in between. This is currently hard coded but the user can apply weights to easily strengthen or weaken it in order to fit personal needs. In principle the polygon itself can be edited and the apk rebuilt if a different shape is required. Developing a GUI for that purpose was considered very tedious especially before knowing whether the results warrant the effort. With this approach you could even approximate the formula well enough that is used in DynamicISF for the ISF dependency on glucose.



There are two weighting factors depending on whether glucose is below or above target:

lower_ISFrange_weight	Used below target, weakens ISF the more the higher this weight is; 0 disables this contribution, i.e. ISF is constant in the whole range below target. This weight is less critical as the loop is probably running at TBR=0 anyway and you can start around 0.2.
higher_ISFrange_weight	Used above target, strengthens ISF the more the higher this weight is 0 disables this contribution, i.e. ISF is constant in the whole range above target. Start with a weight of 0.2 and observe the reactions and check the SMB-tab before you increase it with care.

The result is:

bg_ISF = 1 + *xxx*_ISFrange_weight * glucose_polygon_Lookup

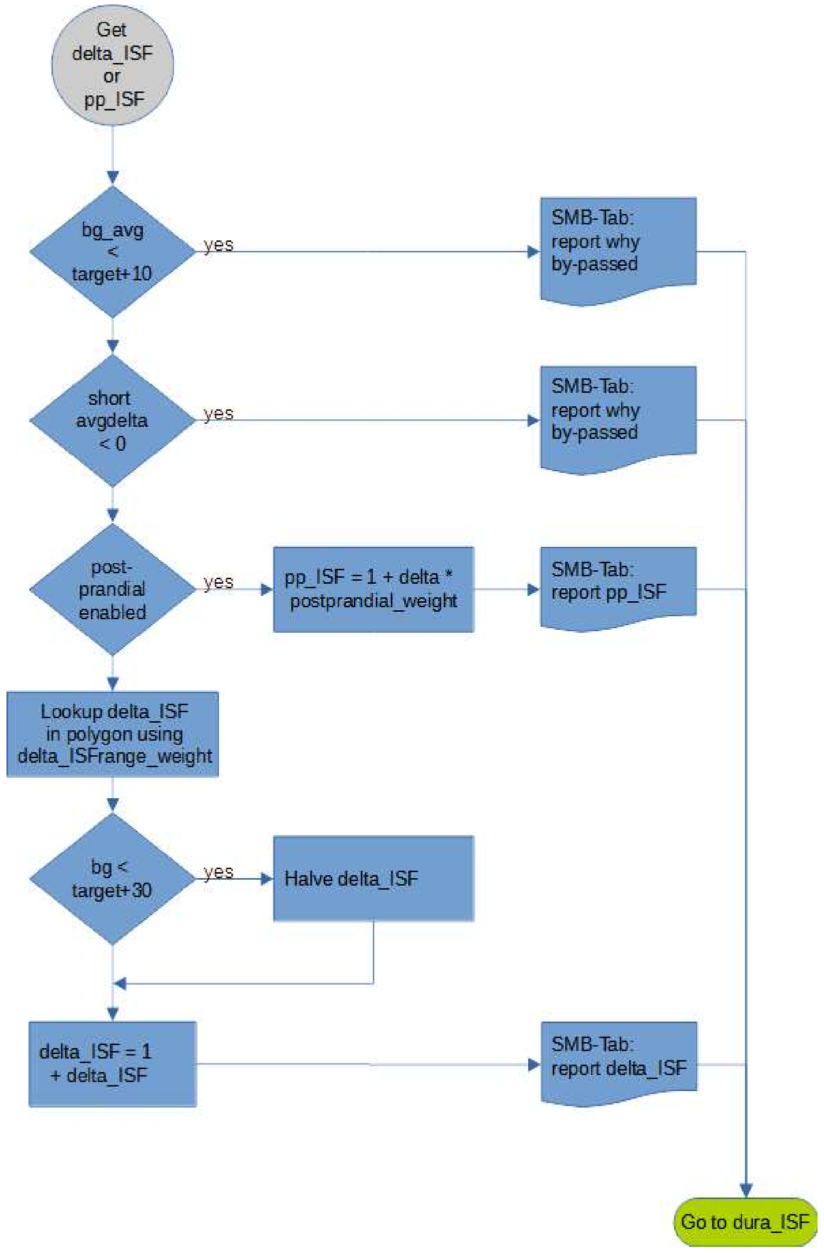
There is a special case possible, namely below target i.e. when bg_ISF<1. ISF will be weakened and there is no point in checking the remaining effects. Only with positive acceleration the weakening will be less pronounced as that is a sign of rising glucose to come soon.

pp_ISF and delta_ISF determination and its impact

There are two alternative effects in autoISF based on delta. The first one becomes active during meal absorption and was introduced to help users with gastroparesis. They can define a time window (*pp_ISF_hours*) during which this effect is active. Alternatively any user can set *enable_pp_ISF_always*=true to activate it all the time. The latter is also the recommended setting for users in pure UAM mode because in their case no meal start can be detected. Given a positive *short_avgdelta* and average glucose being above target+10 the result is:

pp_ISF = 1 +
delta * pp_ISF_weight.

As a starting value for *pp_ISF_weight* use 0.005. Observe the reactions and check the SMB-tab before you increase it with care. A weight of 0 disables this contribution.



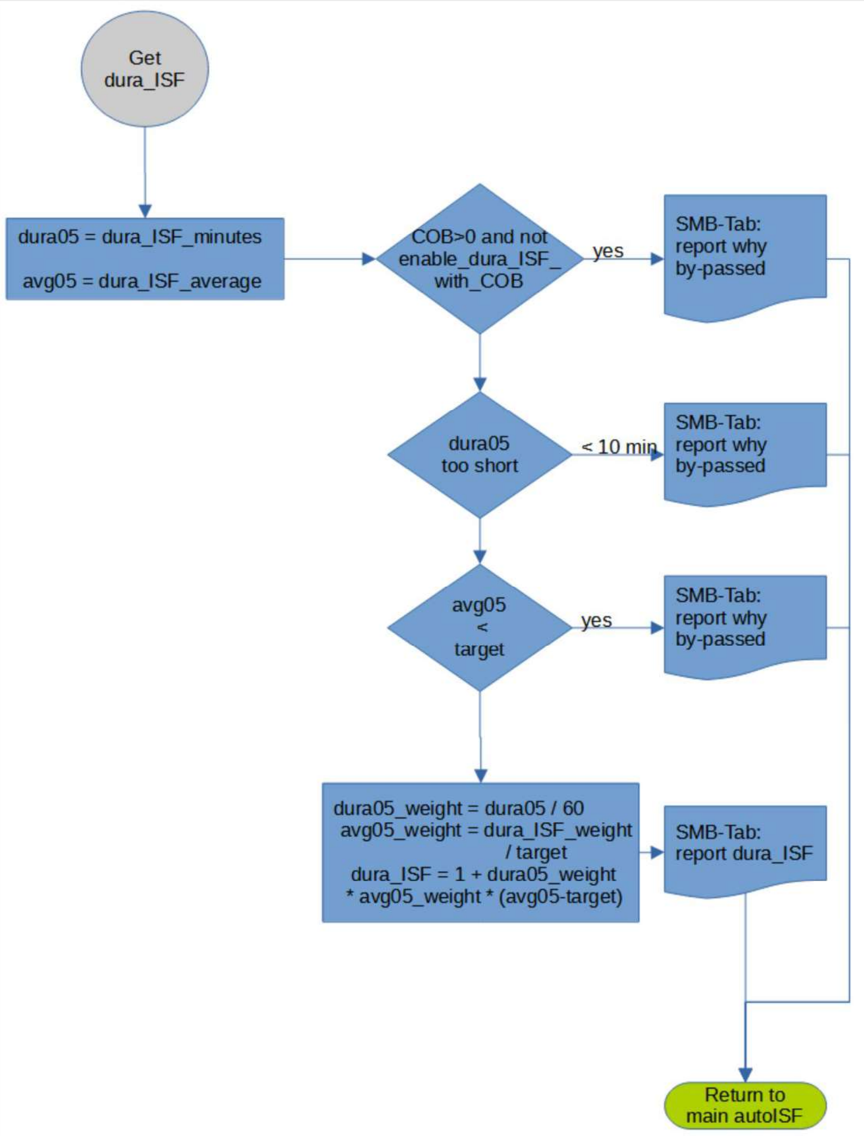
If postprandial settings are not enabled then the alternative method mimics what AAPS users did in defining automation rules that modify profiles or targets based on the level if delta. As in the case of bg_ISF again a polygon is used to lookup an ISF modification. For safety reasons this modification is halved (target_penalty) if bg<target+30. Finally the *delta_ISFrange_weight* is applied:

POLYGON x_data (delta)	POLYGON y_data (Lookup)	delta_ISF with weight=1.0	delta_ISF with weight=0.06
2	0	1	1,000
7	0	1	1,000
12	0,4	1,4	1,024
16	0,7	1,7	1,042
20	0,7	1,7	1,042

delta_ISF = 1 + *delta_ISFrange_weight* * *target_penalty* * *delta_polygon_Lookup*

As a starting value for *delta_ISFrange_weight* use 0.02. Observe the reactions and check the SMB-tab before you increase it with care. A weight of 0 disables this contribution.

dura_ISF determination and its impact



This is the original effect of autoISF in action since August 2020. Because autoISF is now a toolbox of several effects this original effect was renamed *dura_*… It addresses situations when

- glucose is varying within a +/-5% interval only;
- the average glucose (*dura_ISF_average)* within that interval is above target;
- this situation lasted at least for the last 10 minutes (*dura_ISF_minutes*).

This is a classical insulin resistance and is typically caused by free fatty acids which grab available insulin before glucose can. Quite often user get impatient in such a situation and administer one or even more rogue boluses. Again and again that leads to hypos later which the dura_ISF approach avoids if carefully tuned.

The method is active if

- no COB is detected like long-time after a meal or in pure UAM mode;
- or *enable_dura_ISF_with_COB*=true, i.e. even while a meal is still being absorbed.

The strengthening if ISF is stronger the longer the situation lasts and the higher the average glucose is above target:

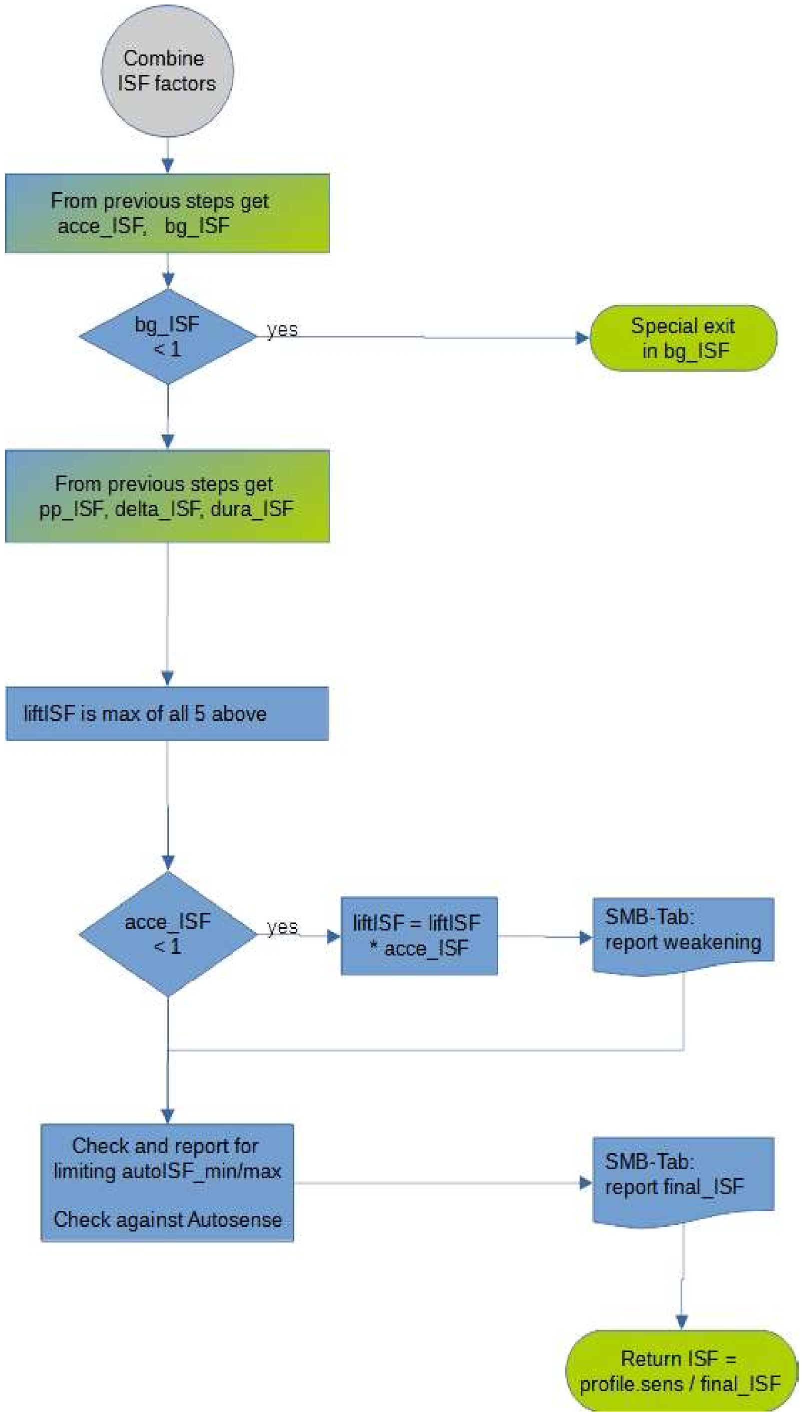
$$\text{dura\_ISF} = 1 + \frac{\text{avg05-target\_bg}}{\text{target\_bg}} * \frac{\text{dura05}}{60} * \text{dura\_ISF\_weight}$$

where

avg05 = dura_ISF_average
dura05 = dura_ISF_minutes

The user can apply his personal weighting by using *dura_ISF_weight*. Start cautiously with a value of 0.2 and be very careful when you approach 1.5 or even higher. By using 0 this effect is disabled.

The combined result from the above factors



Now that the factors for all 4 effects are known how to deduce an end result? The normal case is to pick the strongest factor as the one and only factor to be applied. Here autosense is also part of the game. But how about the exceptions, i.e. when different factors pull in different directions? In order of precedence they are:

- bg_ISF < 1, i.e. glucose is below target
  If acce_ISF>1, i.e. glucose is accelerating although below target, both factors get multiplied as a trade-off between them. Then the weaker of bg_ISF and Autosens is used as the final sensitivity ISF.

- acce_ISF < 1, i.e. glucose is decelerating while other effects want to strengthen ISF.
  In this case the strongest of the remaining, positive factors will be multiplied by acce_ISF to reach a compromise. This overall factor will be compared with autosense and the stronger of the two will be used in calculating the final sensitivity ISF.

In all of the above the autoISF limits for maximum and minimum changes will also be applied.

Exercise mode

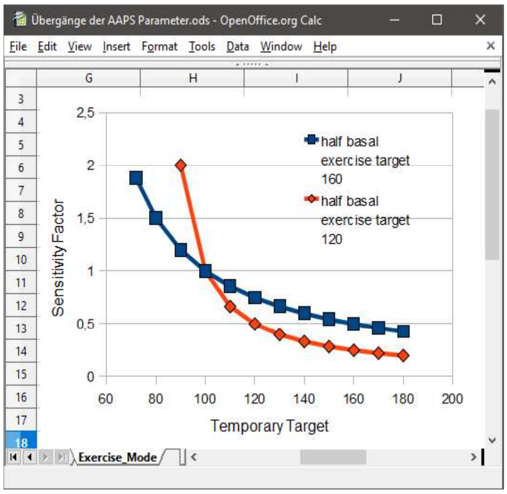
Why can’t I use Exercise mode in standard AAPS but with autoISF?
Exercise mode was disabled in AAPS years ago because there is some risk. Here it is enabled - as it is in OpenAPS too - because it is a powerful tool, but should be used with caution. Scott Leibrandt once described the risks as follows:

"It was a rather rare case where the sensitivity could be too high. It was really a very rare case, you had to meet all of the following 5 conditions.
(Number 5 is a safety setting! You have to have played around with the default settings).

1. *Oref1 sensitivity enabled*
2. *High temptarget raises sensitivity enabled.*
3. *A site change or profile change has been logged within the last few minutes.*
4. *A high temptarget is enabled.*
5. *An unreasonable autosens threshold has been set."*

The toggle on the overview screen was added to quickly deactivate (top) or activate (bottom) the exercise mode. Otherwise the settings can also be adapted individually in the OpenAPS SMB menu. The relevant settings for exercise mode are:

- *high_temptarget_raises_sensitivity* (synonym for exercise mode); defaults to false.
  When set to true, temp targets > 100 raise sensitivity (lower sensitivity ratio/ISF numbers) and reduce basal. The higher your temp target is above 100 the more sensitive the ratios will be, e.g. temp target of 120 results in sensitivity ratio of 0.75, while 140 results in 0.6 (with default *half_basal_exercise_target* of 160). Basal will be reduced accordingly. See also *half_basal_exercise_target.*
- *half_basal_exercise_target*; defaults to 160.
  This means means when temp target is 160 mg/dl and *high_temptarget_raises_sensitivity* =true then run at 50% basal (for TT=120 at 75%; for TT=140 at 60%). This base exercise target number can be adjusted in the OpenAPS SMB menu to give you more control over your exercise modes. In addition to basal it also influences sensitivity. See also *high_temptarget_raises_sensitivity.*



- *low_temptarget_lowers_sensitivity*; defaults to false.
  When set to true it can lower sensitivity (higher sensitivity ratio/ISF numbers) for temp targets < 100. The lower your temp target is below 100 the less sensitive the ratios will be, e.g. temp target of 95 results in sensitivity ratio of 1.09, while 85 results in 1.33 (with default *half_basal_exercise_target* of 160).
  Here, too, the basal is also adjusted and increased accordingly.

Please note
The sensitivity ratio (ISF) modified by exercise mode is the basis for further sensitivity modifications by autoISF. This is an extension to the basic rule that only the stronger of ISF modifiers prevails.

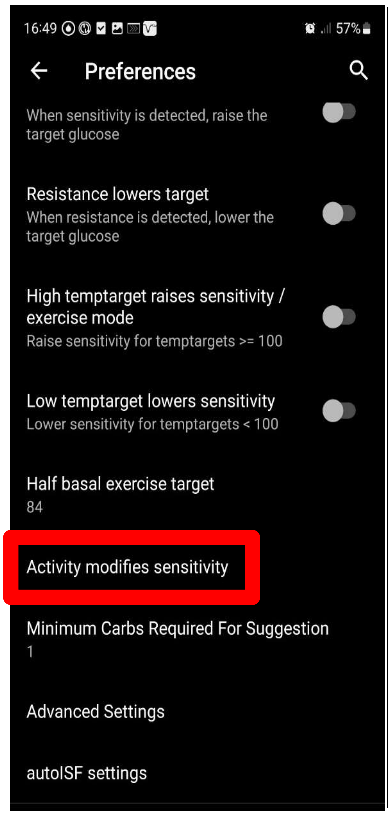
Activity Monitor

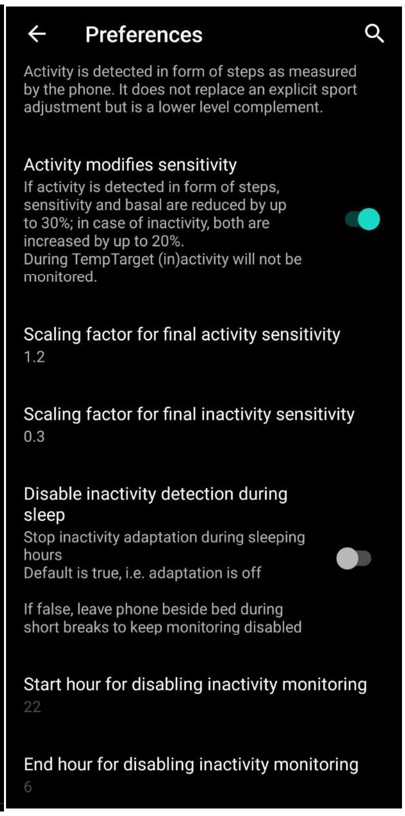
This is a milder method of adapting sensitivity compared to the exercise mode. Examples of when it appears to work well is for vacuum cleaning, a short walk to the cinema or a short bicycle ride for shopping. The base information comes from the phones acceleration sensor and its inbuilt step counter. There is a new switch in the OpenAPS SMB menu to enable the activity monitor. By default it remains inactive. The steps counted can be checked in the SMB tab towards the end of the profile section. The steps are evaluated for various time segments during the last hour and lead to these 5 classifications:

Classification	Description	Max 1.5	Default 1.0	Example 0.6	Min 0.0
activity	step count fairly above neutral	0.55	0.70	0.82	1.00
partial activity	step count somewhat above neutral	0.775	0.85	0.94	1.00
neutral	neutral step count, loop tuned accordingly	1.00	1.00	1.00	1.00
partial inactivity	step count somewhat below neutral	1.15	1.10	1.06	1.00
inactivity	step count fairly below neutral	1.30	1.20	1.12	1.00

The columns to the right show resulting sensitivity change factors for various scale factors. These scale factors modify the default and can be set in the preferences to tune the impact to your personal needs. Right at the start of the SMB-tab debug section the activity monitor lists the current status and assessment of activity level.

Besides the sensitivity the basal rate is also adapted. The one unexpected but welcome side effect of the 1 hour time window holding step counts is the gradual phasing out after the end of activity from 30% via 15% to eventually 0% impact. This also works at the end of exercise mode because activity monitor takes over once the exercise TT ends.





The activity monitor is disabled or limited in either of the following situations:

- no inactivity detection during the first hour of (re-)starting AAPS
- deactivated in settings
- a TT is active
- the phone was not carried during the last 15 minutes

This last exception leads to the question what to do during sleeping hours. With the phone being at rest how can you avoid ending up in inactivity mode if sleep is interrupted and you quickly check the phone or go to the bathroom?

Option 1: just leave the phone next to the bed and don’t touch it; trust the loop or check status on the watch.

Option 2: enable and define your personal window of sleeping hours to disable the inactivity detection.

Once you get up for good you definitely start the day in inactivity mode. This is a welcome state to fight the resistance during the dawn phenomenon.

Internal automation for iobTH

The new variable *iob_threshold_percent* holds a percentage of the max_iob which is used as the threshold to disable SMB. Inside the autoISF code it offers more flexibility than can be achieved with regular AAPS automations used before. The result of any sensitivity change defined by the user is a modulated value labelled internally as effective iobTH.

The new capabilities are:

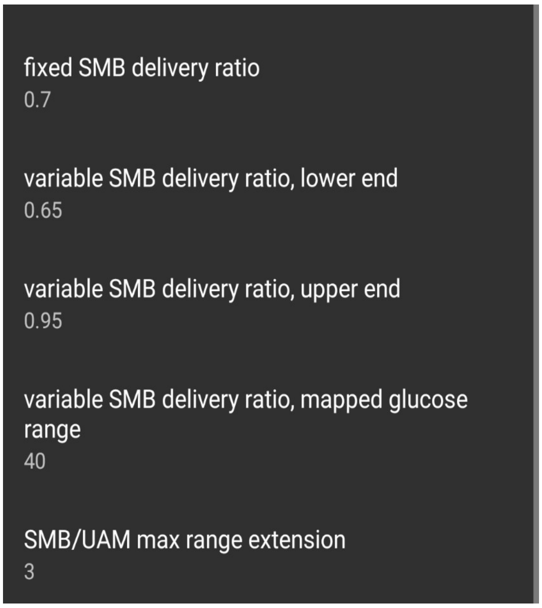
- *iob_threshold_percent* gets modulated while the pump profile is set to a percentage. The idea is that with changed sensitivity the threshold changes accordingly. So internally an effective iobTH is used. If for example the profile is raised to 120% because of an infection then the effective iobTH is 120% of *iob_threshold_percent*. This relieves the user from having to adapt the automation rules for those periods and having to remember setting them back once the profile is reset.
- *iob_threshold_percent* gets modulated while exercise mode is active which implicitly changes sensitivity. The reasoning and rules are the same as in the preceding situation. This modulation is combined with the above profile based modulation.
- *iob_threshold_percent* gets modulated while the activity monitor changes sensitivity. The reasoning and rules are the same as in the preceding situations. This modulation is combined with the above profile based modulation.
- A very special modification happens during the initial rise after carbs intake. After the first few SMBs the iob threshold may eventually be surpassed. Often this initial overshoot was far too much due to limited capabilities using automations and led to hypo later. The new code will limit this overshoot or tolerance to 130% of the effective iobTH.

You may want to consider raising your “classical” iob threshold values used by automations because the new 130% overrun will end up in lower values on average. But then the 130% is not reached every time and raising them by 10%-20% may be a good initial compromise.

The value *iob_threshold_percent* is accessible for manual editing or inspection in the autoISF Menu system under Full Loop.

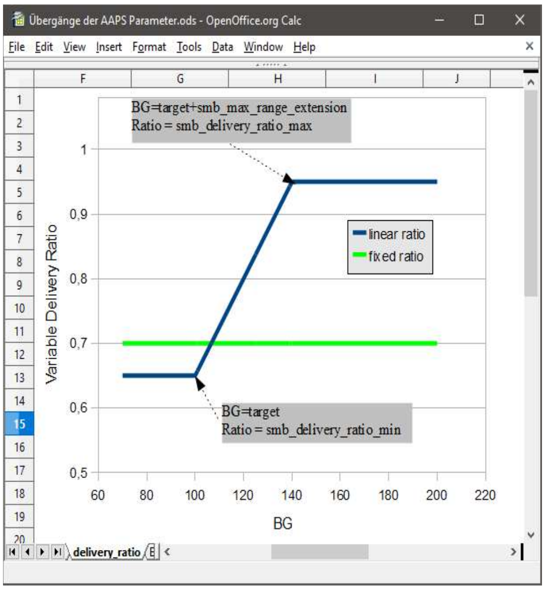
Adapting SMB delivery

After reading all the methods to strengthen ISF you may wonder whether your maxBolus will always allow as much SMB as requested by the loop with such stronger ISF. This is especially important for pure UAM mode, where you want a few but strong SMBs as soon as meal absorption is detected by acce_ISF in order to catch up with pre-bolus and meal bolus in standard use of AAPS. Several extensions in autoISF can be used to get there:

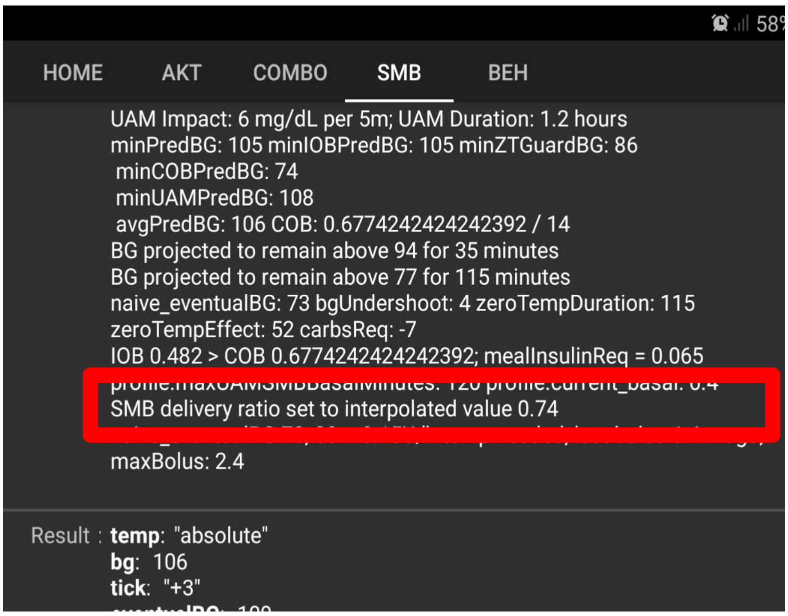


- In AAPS *smb_delivery_ratio* is normally hard coded as 0.5 of the insulin requested. This is a safety feature for master/follower setups in case both phones trigger an SMB in the same situation. If this does not apply in your case you may increase this setting to a value above 0.5 and up to even 1.0 if you are very courageous. Best is to leave some margin like the max delivery setting in the calculator.

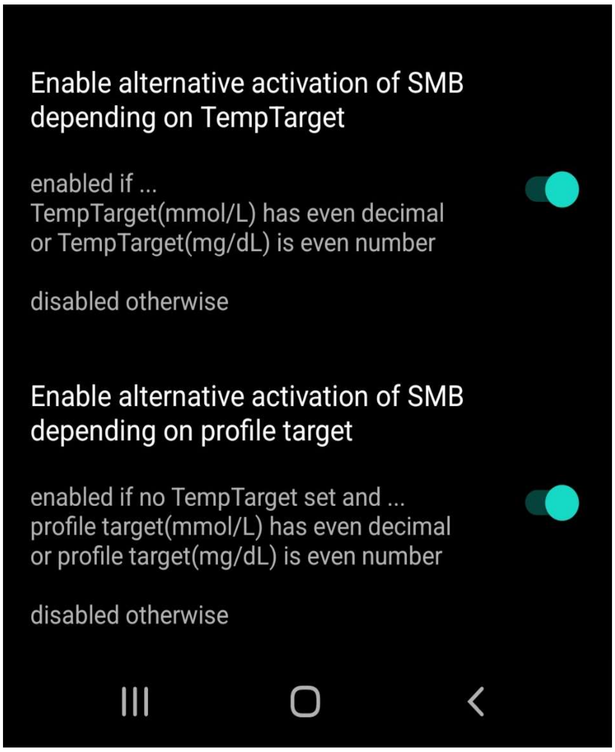
- Alternatively to a higher but fixed ratio you can use a linearly rising ratio, starting cautiously with *smb_delivery_ratio_min* at *target_bg* and rising to a more ambitious *smb_delivery_ratio_max* at *target_bg+smb_delivery_ratio_bg_range*. If *smb_delivery_ratio_bg_range=0* then this linear rise is disabled and the above *smb_delivery_ratio* is used instead.



- smb_max_range_extension is a factor that multiplies the current *maxSMBBasalMinutes* and *maxUAMSMBBasalMinutes* beyond the 120 minute limit.



Alternatively enable SMB depending on target value type



This new feature works independent of other autoISF settings. By clever selection of *Temporary Targets* or *profile targets* you now have additional options for enabling or disabling SMB essentially across the whole allowed range of 72-180mg/dl. The decision is based on whether the target value is an even or odd number. For systems using mmol/l it is slightly adapted in that the decimal figure is even as in 5.2 or odd as in 5.1.

- If the TT is an even number then SMB is always enabled and a message like "*SMB enabled; TempTarget 78 is even number*" or "*SMB enabled; TempTarget 5.2 has even decimal*" shows up in the SMB-tab. This is useful for *Eating Soon* and compatible with its standard assignment of 72. Also, you could for instance select 120 while sick in bed and still get SMBs, regardless of all the other SMB settings.
  The only exceptions are those situations where SMBs are disabled for reasons other than direct SMB preference settings, e.g. a hypo looming in the predictions.
- If the TT is an odd number then SMB is disabled always and a message like "*SMB disabled; TempTarget 81 is odd number*" or "*SMB disabled; TempTarget 5.1 has odd decimal*" shows up in the SMB-tab. This is useful for quiet times or overnight with smoother curves by selecting TT=81 or 83 which worked very well for me. You can also use it overnight to avoid overreactions against compression lows.
- If no TT is set then the equivalent check is done based on the targets defined in the profile. This is especially attractive to disable SMB during sleep times by default with an odd value in the profile. Please note, that to be used the lower and upper targets in the profile need to be identical.
- If none of the above applies or none of the features is enabled then the normal AAPS rules and messages will apply.

You need to be a careful because of **old habits** when defining a TT. *Eating Soon* at TT=72 behaves as before but *Hypo Target* at TT=120 would enable SMB which is probably not what you want in that situation. So better go to settings for *Default Temp-Targets* and change the *hypo target* to 121. You should also check the defaults for *Activity Target* and make sure that fits your traditional SMB option.

With **automations** this offers a wide range of TTs and options without the need to adhere to the watershed at 100mg/dl. As an example take the situation where your IOB gets too high but the carbs are still coming in you can set a TT=73 by automation which gives the strongest possible TBR action but no SMB. **You must check all existing rules that set a TT whether they must be adapted.**

As you can see these new options are powerful but need careful preparation. Therefore the SMB delivery settings menu was extended as shown above by the new switches ***enableSMB_EvenOn_OddOff*** for *Temporary Target*s and ***enableSMB_EvenOn_OddOff_always*** for profile targets to consciously enable this behaviour. Otherwise users not aware of this might get caught out.

New automations specific to the 5 glucose ISF weights

Following requests from several users some automations were developed and added with relevant autoISF usage in mind.

Enable glucose ISF weights – Disable glucose ISF weights

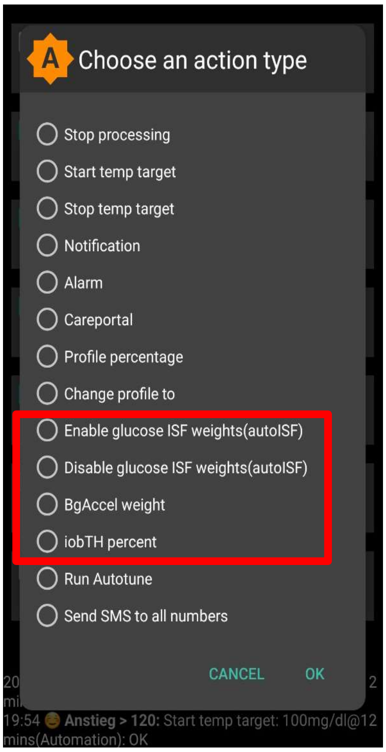
This will set the variable *enable_autoISF* accordingly. It only affects the 5 weights related to glucose behaviour and none of the other features like those influencing SMB.

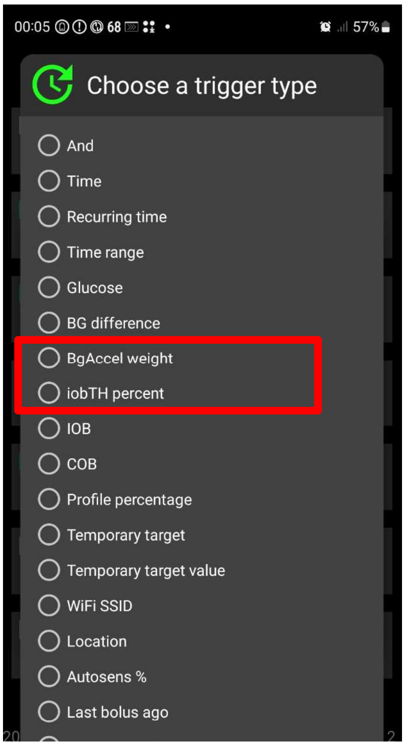
Set BgAccel weight – Set iobTH percent

This will set the two variables. For details of the new variable *iob_threshold_percentage* see the separate topic in this document.

Trigger on BgAccel weight - Trigger on iobTH percent

You can create your usual conditions like "equals" or "less than" against the current value of any of those two variables.

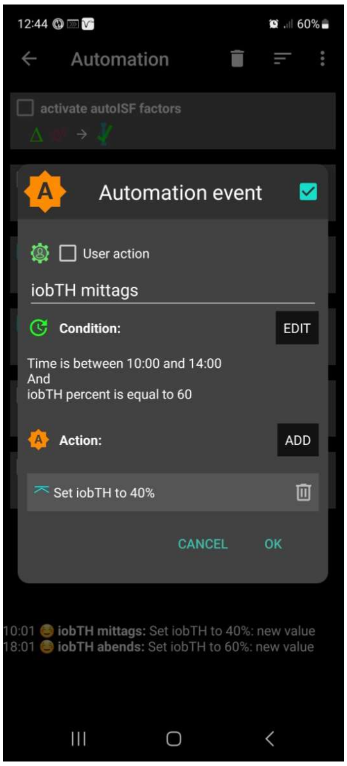


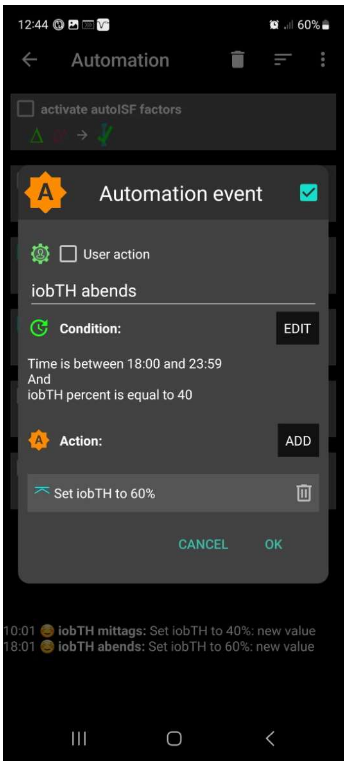


Here is an example set of automations to alternate between two values of iobTH:

I use two different values of *iob_threshold_percent* during a normal day. It is 40% for lunch time and 60% for dinner time. I have these two rules to switch by time of day and only if the current value equals the value from the earlier shift. Any other value is treated as a manual override for special occasions until I manually set it to its regular value. The time windows for switching are long enough to catch an opportunity to be processed and do not need to be actioned half a day each.

The often mentioned extra load on the phone battery is acceptable to me. With the automation plugin disabled completely a full charge lasts nearly 40 hours, with these two active it still lasts nearly 30 hours.





Attachment: Table of all autoISF settings

Name	Use	Min – Max	Default	Useful initial value for adults[1]
Settings related to the 5 glucose ISF weights				
enable_autoISF	Use any of the 5 glucose weights for scaling the glucose ISF factors	False - True	False	
autoISF_min	Lowest ISF factor allowed	0.3 – 1.0	1	0.7 like autosense_min
autoISF_max	Highest ISF factor allowed	1.0 – 3.0	1	1.2 like autosense_max
Settings related to acce_ISF, i.e. related to glucose acceleration				
bgAccel_ISF_weight	Strength of acce_ISF contribution with positive acceleration	0.0 – 1.0	0	0.02
bgBrake_ISF_weight	Strength of acce_ISF contribution with negative acceleration	0.0 – 1.0	0	0.02
Settings related to pp_ISF and delta_ISF, i.e. glucose delta				
enable_pp_ISF_always	Flag to use it even without COB	False - True	False	True for UAM mode
pp_ISF_hours	How many hours after start of a meal the effect is active	1 – 10	3	6 for gastroparesis
pp_ISF_weight	Strength of effect	0.0 – 1.0	0	0.005
delta_ISFrange_weight	Strength of effect outside the above postprandial time window	0.0 – 1.0	0	0.02
Settings related to bg_ISF, i.e. glucose deviating from target				
lower_ISFrange_weight	Strength of bg_ISF effect when bg<target	0.0 – 2.0	0	0.2
higher_ISFrange_weight	Strength of bg_ISF effect when bg>target	0.0 – 2.0	0	0.2
Settings related to dura_ISF, i.e. related to glucose “frozen” at high levels				
enable_dura_ISF_with_COB	Enable dura_ISF even if COB is present	False - True	False	
dura_ISF_weight	Strength of dura_ISF effect	0.0 – 3.0	0	0.2
Settings related to SMB delivery size				
smb_delivery_ratio	Increase the AAPS standard 0.5 of Insulin required	0.5 – 1.0	0.5	0.6
smb_delivery_ratio_min	Minimum for linearly rising ratio at and below target_bg	0.5 – 1.0	0.5	0.5
smb_delivery_ratio_max	Maximum for linearly rising ratio at and above target_bg + smb_delivery_ratio_bg_range	0.5 – 1.0	0.9	0.8
smb_delivery_ratio_bg_range	Width of bg range to reach maximum ratio	0.0 – 100	0	40
smb_max_range_extension	Multiplier for maxSMBBasalMinutes and maxUAMSMBBasalMinutes	1.0 – 5.0	1	1.5
Settings related to enabling/disabling SMB by even or odd target values				
enableSMB_EvenOn_OddOff	Enable SMB depending on TT value type	False - True	False	
enableSMB_EvenOn_OddOff_always	Enable SMB depending on profile target value type	False - True	False	
Settings related to scaling the activity monitor impact				
activity_scale_factor	Strength of effect	0.0 – 1.5	1.0	
inactivity_scale_factor	Strength of effect	0.0 – 1.5	1.0	

1 For children there is minimal experience and you need to use much softer settings

Attachment: Screenshot of autoISF menu