Fair Usage Policy (FUP)

Introduction

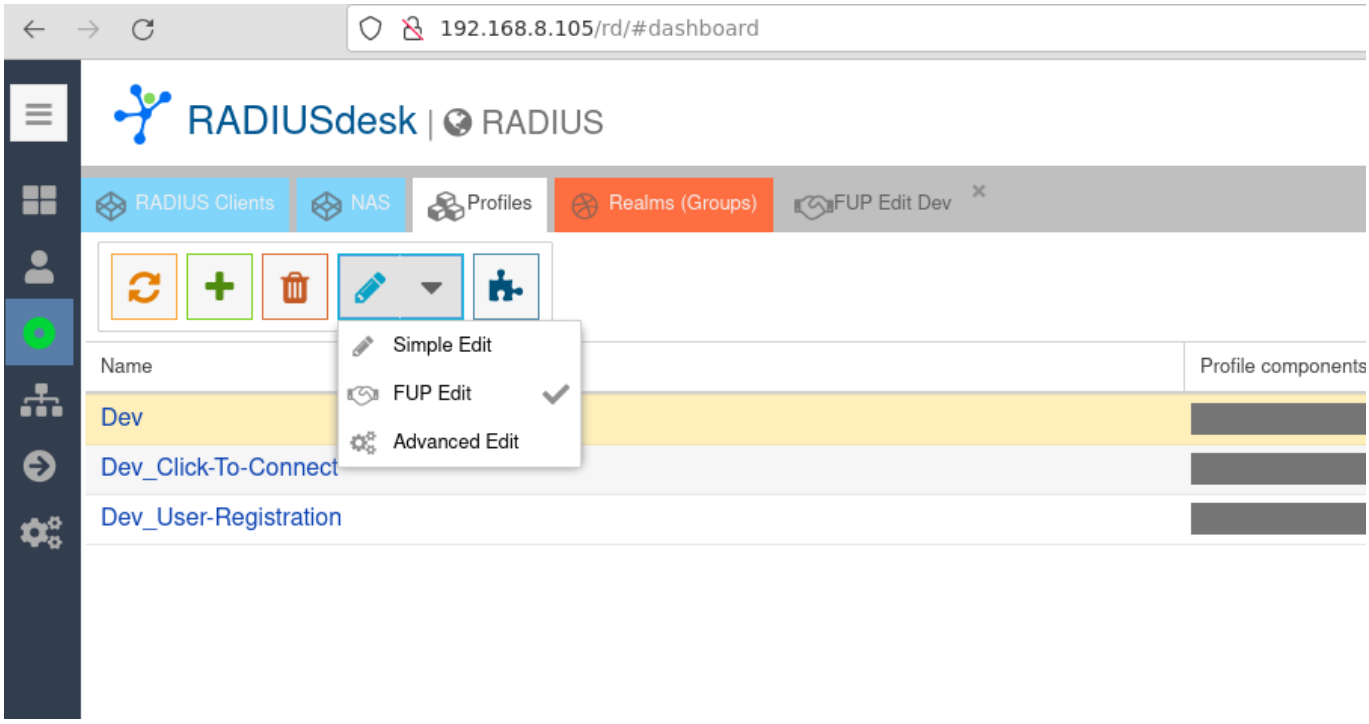
- From January 2023 RADIUSdesk now also includes a powerful FUP package.
- We worked with our clients to come up with an innovative and flexible implementation that packs a punch.
- This document will start with a high level discussion about the various FUP requirements met with this implementation.
- It will then go on to do a hands-on FUP profile.
- Finally we will show you where to tweak things should the need arise for your specific environment or implementation.

FUP Implementation

- In South Africa some of the big ISPs implement multiple layers of service for their FUP.
    - There will be a reduction in bandwidth when a certain amount of data is used during the month.
    - There will be a further reduction in bandwidth when a second milestone of data used is reached.
    - Finally when a third milestone is reached the bandwidth is throttled down to a trickle.
    - When the new month starts everything is reset back to normal speed again.
- Another requirement is the ability to assign an IP Pool to a certain level of service.
    - This is typically with ISPs that use Mikrotik PPPoE servers.
    - There will for instance be a premium IP Pool and a best effort IP Pool.
    - A user will start off on the premium IP Pool up to the point where their specified FUP is triggered.
    - Thereafter they are moved to the best effort IP Pool.
- Communities that uses RADIUSdesk wanted to offer their users more bandwidth between midnight and 7AM since the utilization on the up-link are very low during that time and it can then encourage a more distributed usage graph.
- With these various requirements in mind we formulated the FUP package in RADIUSdesk.

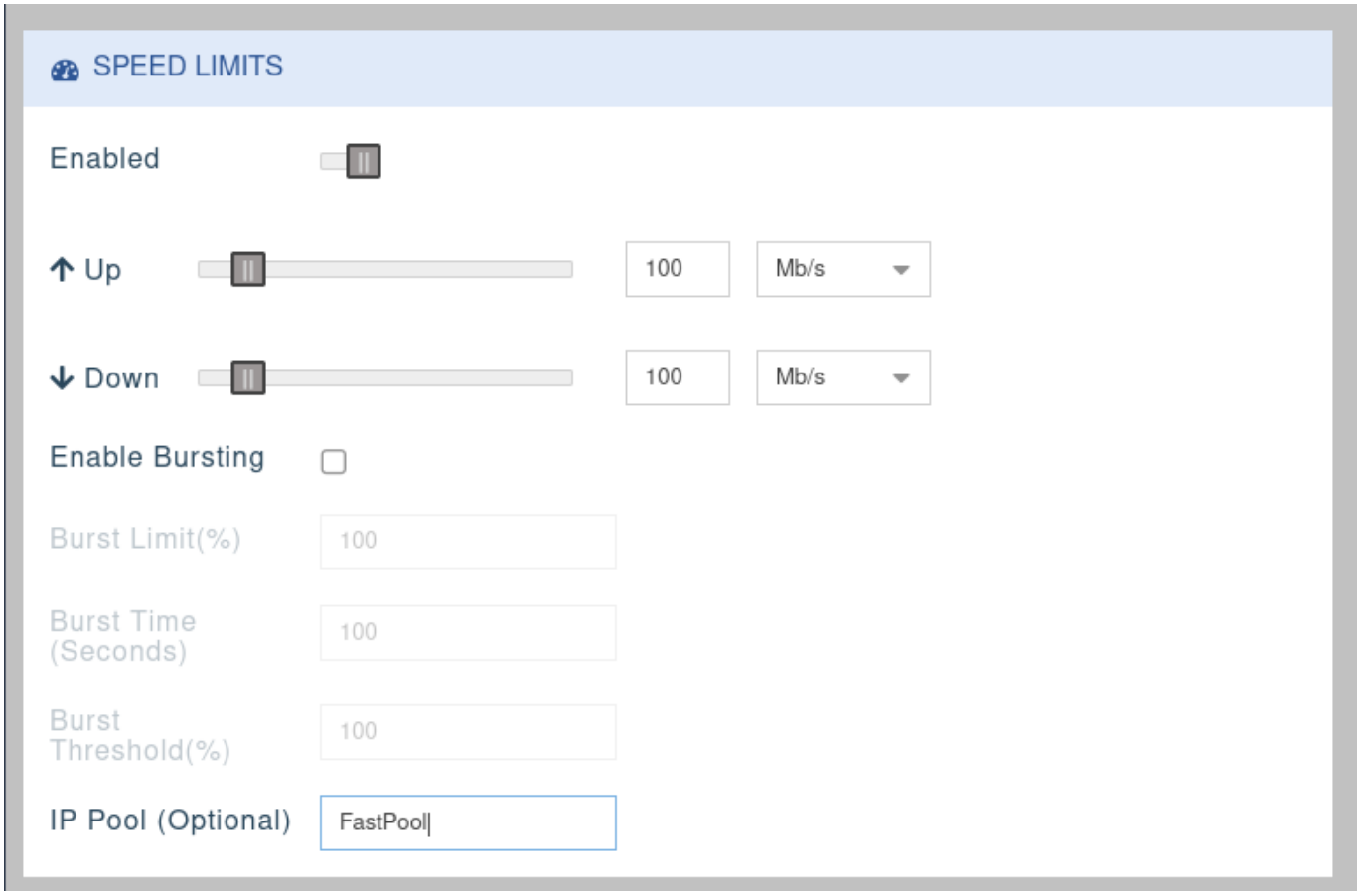
Hands-on FUP Profiles

- To configure the FUP part of a profile, go to **RADIUS → Profiles**.
- The edit option of a profile includes **FUP**.

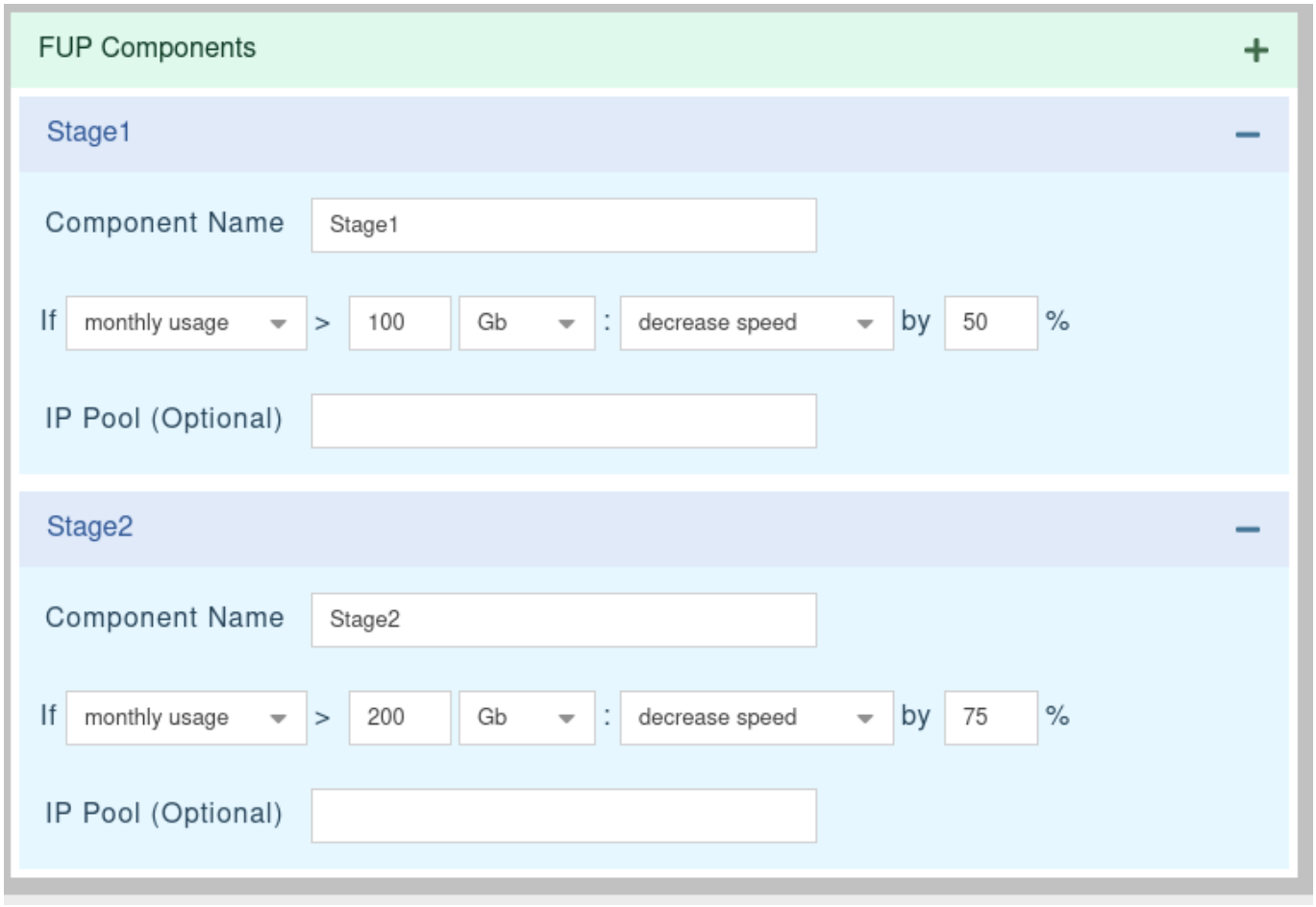


- There are two sections for a FUP based profile.
  - The first section specifies the basic speed when no limits are imposed.
  - The second section contains a list of restriction FUP components to apply.

Basic Speed Limit - No limits imposed



FUP Components to apply



- In the screenshot above we see two stage throttling.
- When the user's total usage reached 100Gb we will half its bandwidth. (reduce to 50Mb/s)
- Then when the user reaches a usage of 200Gb we will reduce its bandwidth further. (reduce to 25Mb/s)
- The FUP component is composed of the following items.
  - Descriptive name.
  - **If condition**. Options include **Time of day**, **Daily usage**, **Weekly usage**, **Monthly usage**.
  - For **Time of day** there will be a start time and end time.
  - For **data usage** there will be an amount of data.
  - **Action**. When the trigger has been reached we can **block the traffic**, **decrease the speed** or **increase the speed**.
  - An optional IP Pool that should be used if the component is triggered.
- You can combine various type of FUP components together.
  - The logic that applies them will use the following rules.
    - Blocking traffic will take preference.
    - Components that decrease the speed will use the slowest (biggest decrease percentage)
    - Components that increase the speed will use the slowest (smallest increase percentage)
  - This essentially means the strictest component's action will be applied.

Important points on FUP

- For FUP to work correct there are two important items which has to be in place and work.

Timezone setting on RADIUS Client

- To determine the exact start of day, week or month the timezone that a RADIUS Client is deployed in needs to be specified.
- If this is not set it will fall back to the timezone that the RADIUSdesk installation is set to.

Ability to disconnect a RADIUS user from RADIUSdesk

- In order for the system to detect and activate an FUP restriction it needs to disconnect an active session of a user.
- The RADIUS Client should then re-authenticate the client where the restriction will be applied. (This is standard procedure for PPPoE connections)

Nuts and bolts of FUP

- This section will be for the readers that likes to know how everything works and fits together.
- When you use the FUP editor for a RADIUS Profile a few things happens behind the scenes.
- The system creates a Profile Component with naming the convention starting with **FupAdd_<profile_id>** and adds this Profile Component to the Profile.
- This Profile Component contains one or more of the following FreeRADIUS custom check attributes.

```
ATTRIBUTE Rd-Fup-Bw-Up           3166 integer
ATTRIBUTE Rd-Fup-Bw-Down         3167 integer
ATTRIBUTE Rd-Fup-Comp-Count      3168 integer
ATTRIBUTE Rd-Fup-Profile-Id      3169 integer
ATTRIBUTE Rd-Fup-Burst-Limit     3170 integer
ATTRIBUTE Rd-Fup-Burst-Time      3171 integer
ATTRIBUTE Rd-Fup-Burst-Threshold 3172 integer
ATTRIBUTE Rd-Fup-Ip-Pool         3173 string
```

- FreeRADIUS is configured to use a combination of **unlang** and a Perl module to formulate the reply to an Access Request that a RADIUS Client will sent to the FreeRADIUS server.
- The Perl module will look for any entries in the **profile_fup_components** DB table that is associated with the RADIUS Profile.
- It will then try and determine which restriction to apply, if any.
- The Perl module can be found in */etc/freeradius/3.0/mods-config/perl/fup.pl*
- It connects to the database and the credentials to connect with to the database is also specified inside this file.
- Should there a FUP component to be applied to a user, we will keep track of it in the **applied_fup_components** table.
- We then run a cron script **cd /var/www/html/cake4/rd_cake && bin/cake fup** that will compare the applied FUP component with the current active FUP component.
- If they are different we send a disconnect request to all the RADIUS Clients for that username (All the RADIUS Clients where the user might be connected to)
- This should initiate a re-authentication which will bring the applied FUP component in sync with the current active FUP component.