pfSense - Feature #3623

Allow each package to choose if it is restarted on interface events

04/23/2014 09:24 PM - Phillip Davis

Status:	New	**Start date:**	04/23/2014
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:	Package System	**Estimated time:**	0.00 hour
Target version:			

Description

Example discussion here about LCD-Proc: https://forum.pfsense.org/index.php?topic=7920.msg413642#msg413642

Some packages have code/daemon/processes that ddoes stuff that does not depend on the real-time interface state. Running the package restart code for these every time there is an interface state change (or other "significant" system event) is unnecessary and just results in interruption to service and possible issues for those packages.
Provide a way for a package to indicate/decide if it needs to actually bother restarting.
Possible solution:
a) Leave the "start" and "stop" actions as is - calling /var/local/etc/rc.d/packagename.sh "stop" would always expect the package to cleanup and stop, and similarly calling "start" always expects the package to really start.
b) "restart" action - add an optional parameter 2 (P2) to indicate the restart reason/type. For now, the only "reason" I can think of is "interfacechange". If P2 is not given, then all packages should respond by doing a full stop and start, like they do now. (This allows a manual restart from the webGUI, for example, to be assured that the package will actually stop and start). If P2 is given then each package can choose how to respond, based on the value of P2.
c) Events that currently call each packagename.sh "restart" will now also pass the extra parameter to indicate what type of event it is.

For something like LCD-proc and similar, it can choose to do nothing if P2 is "interfacechange" (or if P2 is anything at all).

History

#1 - 04/24/2014 06:23 AM - Steve Wheeler

Completely behind this.
Would the additional parameter cause an issue for rc scripts that were not written to expect it?

In the LCDproc case restarting the package after boot is not an issue, it restarts fine. The problem is that at boot the package is restarted so many times in such a short space of time that it causes unpredictable results.

I note that some other packages have code to prevent them starting until after boot.

Steve

#2 - 04/26/2014 12:49 PM - Gunther Jauch

Phillip Davis wrote:

> Possible solution:
> a) Leave the "start" and "stop" actions as is - calling /var/local/etc/rc.d/packagename.sh "stop" would always expect the package to cleanup and stop, and similarly calling "start" always expects the package to really start.
> b) "restart" action - add an optional parameter 2 (P2) to indicate the restart reason/type. For now, the only "reason" I can think of is "interfacechange". If P2 is not given, then all packages should respond by doing a full stop and start, like they do now. (This allows a manual restart from the webGUI, for example, to be assured that the package will actually stop and start). If P2 is given then each package can choose how to respond, based on the value of P2.
> c) Events that currently call each packagename.sh "restart" will now also pass the extra parameter to indicate what type of event it is.

There is not really a need for a complete new "restart" action!

It's possible to always send an additional parameter inside rc.start/stop_packages scripts

Right now the following code is used to start a script under /usr/local/etc/rc.d

```
fwrite($shell, "{$rcfile} start >>/tmp/bootup_messages 2>&1 &");
```

you could easily add the second parameter indicating the "reason" for the call there

```
fwrite($shell, "{$rcfile} start $reason >>/tmp/bootup_messages 2>&1 &");
```

e.g. $reason could be "startup", "interfacechange" and so on...

Someone must provide a function that can fill the $reason with the corresponding settings! Then you can simply handle the $reason value via $2 inside the scripts under /usr/local/etc/rc.d

#3 - 04/26/2014 12:53 PM - Gunther Jauch

Phillip Davis wrote:

> Possible solution:
> a) Leave the "start" and "stop" actions as is - calling /var/local/etc/rc.d/packagename.sh "stop" would always expect the package to cleanup and stop, and similarly calling "start" always expects the package to really start.
> b) "restart" action - add an optional parameter 2 (P2) to indicate the restart reason/type. For now, the only "reason" I can think of is "interfacechange". If P2 is not given, then all packages should respond by doing a full stop and start, like they do now. (This allows a manual restart from the webGUI, for example, to be assured that the package will actually stop and start). If P2 is given then each package can choose how to respond, based on the value of P2.
> c) Events that currently call each packagename.sh "restart" will now also pass the extra parameter to indicate what type of event it is.

There is not really a need for a complete new "restart" action!
It's possible to always send an additional parameter inside rc.start/stop_packages scripts

Right now the following code is used to start a script under /usr/local/etc/rc.d

```
fwrite($shell, "{$rcfile} start >>/tmp/bootup_messages 2>&1 &");
```

you could easily add the second parameter indicating the "reason" for the call there

```
fwrite($shell, "{$rcfile} start $reason >>/tmp/bootup_messages 2>&1 &");
```

e.g. $reason could be "startup" (after reboot / normal boot up), "interfacechange" and so on...

Someone must provide a function that can fill the $reason with the corresponding values! Then you can simply handle the $reason value via $2 inside the scripts under /usr/local/etc/rc.d

#4 - 04/26/2014 01:00 PM - Gunther Jauch

A simple working "workaround" approach to notice if a restart of packages occured could be realized like so:

1. The restart itself is initiated via a function inside /etc/inc/util.inc called send_event(). This function open a socket to a daemon and send "commands" to him.
In our case the command "service reload packages"

2. We can intercept/modify this send_event() function to see if a successfull restart of the packages has been triggered and touch an "event file" to indicate this.

3. Now we can check at the beginning of the scripts under /usr/local/etc/rc.d if the "event file" exists and act to it if needed.

4. At the end we need a "delete script" under /usr/local/etc/rc.d that runs after all other scripts and deletes our "event file" to reset our mechanism back to null

More details can be found in my posting: https://forum.pfsense.org/index.php?topic=7920.msg414439#msg414439

cu Gunther