NAME

systemd.offline-updates − Implementation of offline updates in systemd

IMPLEMENTING OFFLINE SYSTEM UPDATES

This man page describes how to implement "offline" system updates with systemd. By "offline" OS updates we mean package installations and updates that are run with the system booted into a special system update mode, in order to avoid problems related to conflicts of libraries and services that are currently running with those on disk. This document is inspired by this **GNOME design whiteboard**[1].

The logic:

1. The package manager prepares system updates by downloading all (RPM or DEB or whatever) packages to update off−line in a special directory /var/lib/system−update (or another directory of the package/upgrade manager's choice).
2. When the user OK'ed the update, the symlink /system−update is created that points to /var/lib/system−update (or wherever the directory with the upgrade files is located) and the system is rebooted. This symlink is in the root directory, since we need to check for it very early at boot, at a time where /var is not available yet.
3. Very early in the new boot **systemd-system-update-generator**(8) checks whether /system−update exists. If so, it (temporarily and for this boot only) redirects (i.e. symlinks) default.target to system−update.target, a special target that pulls in the base system (i.e. sysinit.target, so that all file systems are mounted but little else) and the system update units.
4. The system now continues to boot into default.target, and thus into system−update.target. This target pulls in all system update units. Only one service should perform an update (see the next point), and all the other ones should exit cleanly with a "success" return code and without doing anything. Update services should be ordered after sysinit.target so that the update starts after all file systems have been mounted.
5. As the first step, an update service should check if the /system−update symlink points to the location used by that update service. In case it does not exist or points to a different location, the service must exit without error. It is possible for multiple update services to be installed, and for multiple update services to be launched in parallel, and only the one that corresponds to the tool that *created* the symlink before reboot should perform any actions. It is unsafe to run multiple updates in parallel.
6. The update service should now do its job. If applicable and possible, it should create a file system snapshot, then install all packages. After completion (regardless whether the update succeeded or failed) the machine must be rebooted, for example by calling **systemctl reboot**. In addition, on failure the script should revert to the old file system snapshot (without the symlink).
7. The upgrade scripts should exit only after the update is finished. It is expected that the service which performs the upgrade will cause the machine to reboot after it is done. If the system−update.target is successfully reached, i.e. all update services have run, and the /system−update symlink still exists, it will be removed and the machine rebooted as a safety measure.
8. After a reboot, now that the /system−update symlink is gone, the generator won't redirect default.target anymore and the system now boots into the default target again.

RECOMMENDATIONS

1. To make things a bit more robust we recommend hooking the update script into system−update.target via a .wants/ symlink in the distribution package, rather than depending on **systemctl enable** in the postinst scriptlets of your package. More specifically, for your update script create a .service file, without [Install] section, and then add a symlink like /lib/systemd/system−update.target.wants/foobar.service → ../foobar.service to your package.
2. Make sure to remove the /system−update symlink as early as possible in the update script to avoid reboot loops in case the update fails.

3. Use *FailureAction=reboot* in the service file for your update script to ensure that a reboot is automatically triggered if the update fails. *FailureAction=* makes sure that the specified unit is activated if your script exits uncleanly (by non−zero error code, or signal/coredump). If your script succeeds you should trigger the reboot in your own code, for example by invoking logind's **Reboot()** call or calling **systemctl reboot**. See **logind dbus API**[2] for details.
4. The update service should declare *DefaultDependencies=no*, *Requires=sysinit.target*, *After=sysinit.target*, *After=system−update−pre.target*, *Before=system−update.target* and explicitly pull in any other services it requires.
5. It may be desirable to always run an auxiliary unit when booting into offline−updates mode, which itself does not install updates. To do this create a .service file with *Wants=system−update−pre.target* and *Before=system−update−pre.target* and add a symlink to that file under /lib/systemd/system−update.target.wants .

SEE ALSO

systemd(1), **systemd.generator**(7), **systemd-system-update-generator**(8), **dnf.plugin.system-upgrade**(8)

NOTES

1. GNOME design whiteboard
   https://wiki.gnome.org/Design/OS/SoftwareUpdates
2. logind dbus API
   https://www.freedesktop.org/wiki/Software/systemd/logind