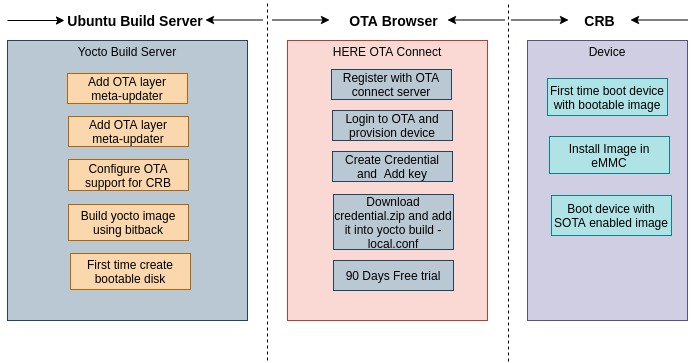
**Intel BSP OTA Support addition in existing Yocto Build environment -**

OTA Design Approach -



The Yocto project meta-updater layer is based on OSTree. The meta-updater layer has included the Intel platform support.

OSTree - OSTree is an upgrade system for Linux-based operating systems that performs atomic upgrades of complete filesystem trees.

OSTree is designed for deploying core systems. The core OSTree model is like git in that it checksums individual files and has a content-addressed-object store.

OSTree Features -

1. OSTree design fully atomic and safe upgrade.

* This means that if an update is not successful, it must not be partially installed. The failure must leave the device in the same state as if the update did not start and no intermediate state must exist

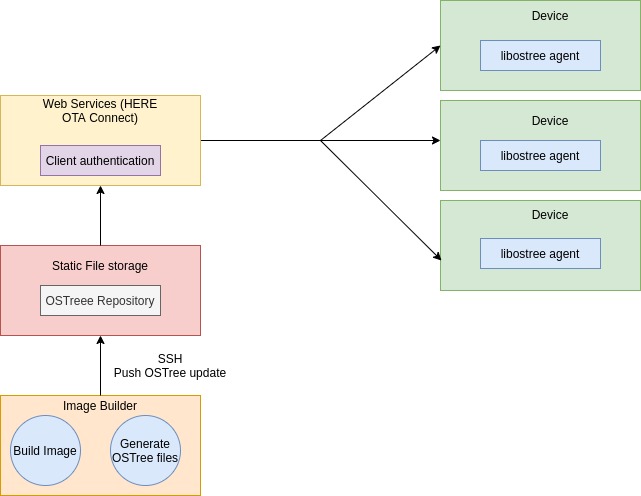
1. OSTree operates at the Unix filesystem layer and thus on top of any filesystem or block storage layout.
2. OSTree does not impose strict requirements on the partitioning scheme
3. Security is at the core of OSTree, offering content replication incrementally over HTTPS via GPG signatures and using SHA256 hash checksums.
4. Support rollback feature - If system is crashed or power pulled, you will have either the old system or new one.
5. Failures during an OSTree atomic update are not committed, meaning that a failed update have no effect on the running system.
6. If an OSTree atomic update completes successfully but introduces software issues, rolling back to the previous working version is guaranteed to work.
7. OSTree, you maintain system images in a Web server-based repository.
8. The images are automatically added to the Web server repository at build time, providing an incremental history - and system fallback - for each build.

Upgrade Workflow -

1. First time system boots with the OTA supported implementation image.
2. A new version is made available as a new OSTree commit in the local repository, either dowloading it from the network.
3. The new version is deployed.
4. The system reboots into the new deployment.
5. If the system fails to boot properly (which should be determined by the system boot logic), the system can roll back to the previous deployment.

Web-Based OTA Update -

1. The image building pipelines pushescommits to an OSTree repository on each build.
2. a standard web server provides access over HTTPS to the OSTree repository handling it as a plain hierarchy of static files, with no special knowledge of OSTree.
3. The client devices poll the web server and retrieve updates when they get published.



**Yocto Build Server**

Setup the Layer -

1. Clone the [meta-updater](https://github.com/advancedtelematic/meta-updater) layer and add it to your [bblayers.conf](https://www.yoctoproject.org/docs/3.0/ref-manual/ref-manual.html#structure-build-conf-bblayers.conf).
2. Clone a BSP integration layer (meta-updater-${PLATFORM}, e.g. [meta-updater-](https://github.com/advancedtelematic/meta-updater-raspberrypi)minnowboard) and add it to your conf/bblayers.conf.
3. To add the support for leafhill need to add the new layer to leafhill or replace the minnowboard with leafhill.
4. Set up your [distro](https://www.yoctoproject.org/docs/3.0/ref-manual/ref-manual.html#var-DISTRO). If you are using "poky", the default distro in Yocto, you can change it in your conf/local.conf to poky-sota or to poky-sota-systemd.
5. [Create a provisioning key](https://connect.ota.here.com/#/profile/access-keys) and add it to your local.conf.

Build Image -

1. Build your image as usual, with bitbake.
2. After building the root file system, bitbake will then create an [OSTree-enabled version](https://ostree.readthedocs.io/en/latest/manual/adapting-existing/) of it, commit it to your local OSTree repo, and push it to OTA Connect.
3. A live disk image will be created (normally named ${IMAGE\_NAME}.wic e.g. core-image-minimal-intel-corei7-64.wic).
4. A local OSTree repository storing all of the filesystem revisions you’ve built.
5. The disk image is what you need to flash onto the target device initially; the OSTree repository is what we use to update the images.
6. OSTree repositories work similar like git repository works.

**HERE OTA Connect -**

OTA Client

1. HERE OTA Connect is highly secure, open, OTA software management solution.
2. It’s designed specially for automotive.
3. Provision a new device.
4. Build a Yocto image that can do atomic full-filesystem updates with rollback.
5. Build a new version of the image and push the update to OTA Connect.
6. Send it to a target device for installation.

Validate Updates -

1. Need to add the feature to auto reboot after successful SW update on target board.
2. Once the target board boots with an updated image, build time and version can be validated.

**Target Device CRB -**

1. First time boot the device with an OTA created disk image.
2. Install the image on eMMC and boot with an OTA supported image.

**SW Update Via Command Line using OSTree -**

1. Install HTTP server on your Ubuntu Linux machine.

$ sudo apt install apache2

$ sudo a2enmod userdir

$ mkdir -p ~/public\_html/ostree/repo

$ sudo systemctl restart apache2.service

1. Set the libostree repository path in your local.conf file.

Add OSTREE\_REPO = "{absolute path of target directory}" to local.conf

Specify a directory under the web root directory, so that it can be accessed via HTTP.

1. Go to Yocto build directory and execute below command

echo "OSTREE\_REPO = \"\${HOME}/public\_html/ostree/repo\"" >> local.conf

1. Build image using bitbake

$ bitbake core-image-minimal

1. Go to you local repo

$ cd /home/vvdn/public\_html/ostree/repo/

1. Run the simple http server in ostree repo

$ **python -m SimpleHTTPServer <port>** # port defaults to 8000

**Use the repository for image upgrade -**

On the target device, take the steps to upgrade the image using libostree.

1. Add the remote repository:

# ostree remote add --no-gpg-verify my\_repo\_2 http://192.168.1.3:8000

1. Verify the remote repository url

# ostree remote show-url my\_repo\_2

1. The above command will display the http remote link.
2. Pull from the remote repository:

ostree pull --commit-metadata-only --depth={depth} {repository name} {branch name}

--commit-metadata-only: don't download actual files

--depth: number of commits to pull. (-1: unlimited)

# ostree pull my\_repo\_2 qemu-ota-project\_test\_cli

ostree pull --commit-metadata-only --depth=3 my\_repo\_2 qemu-ota-project\_test\_cli

ostree pull --commit-metadata-only my\_repo\_2 qemu-ota-project\_test\_cli

1. Check the libostree commit log:

ostree log {repository name}:{branch name}

# ostree log my\_repo\_2:qemu-ota-project\_test\_cli

1. Deploy a specific revision, and reboot the target.

ostree admin switch {repository name}:{commit hash value}

# ostree admin switch my\_repo\_2:edb8e1a964c185d33296e990c1dcd2719aeb8a2696ab665b316ad5c0d2985b73

1. Reboot the target.
2. Check the booted revision:

#ostree admin status

**Reference -**

<https://docs.ota.here.com/getstarted/dev/index.html>

<https://ostree.readthedocs.io/en/latest/manual/introduction/>

<https://www.witekio.com/blog/hands-ostree-differential-system-updates/>

<https://www.webosose.org/docs/guides/setup/setting-up-fota/>

<https://designs.apertis.org/latest/system-updates-and-rollback.html>