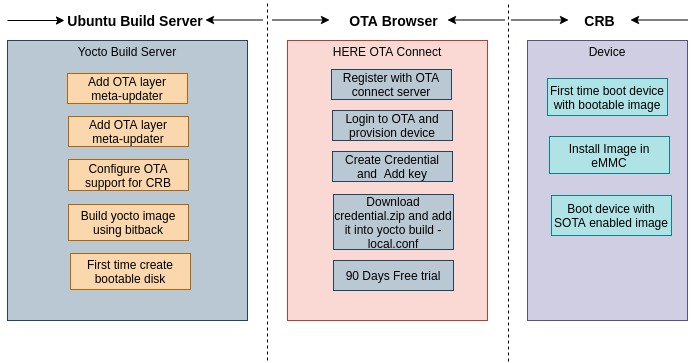
**Intel BSP OTA Support addition in existing Yocto Build environment -**

**Yocto BSP (kernel+rfs) OTA Design Approach -**



**meta-updater Layer for OTA -**

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 -**

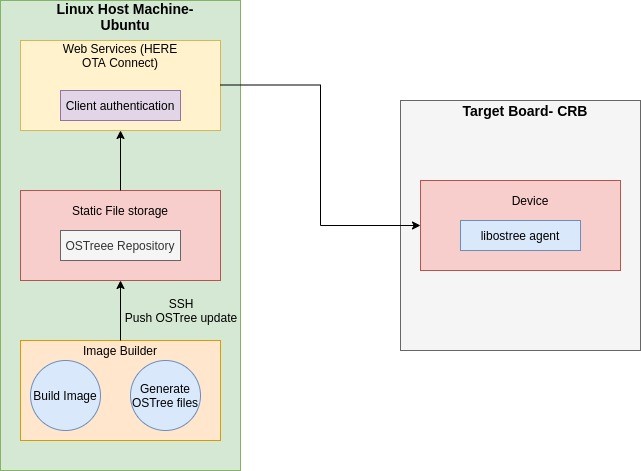
1. OSTree design fully atomic and safe upgrade.
2. OSTree operates at the Unix filesystem layer and thus on top of any filesystem or block storage layout.
3. OSTree does not impose strict requirements on the partitioning scheme
4. Security is at the core of OSTree, offering content replication incrementally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

**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 downloading 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 pushes commits 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 Yocto build 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) on HERE OTA Connect server and add the key to the 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. New version can be verified on HERE OTA connect GUI.
3. 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.
3. Once the device is booted with OTA supported image. The device has the capability to upgrade the new image.
4. Whenever the new changes build, the build image is pushed to HERE OTA Connect server.
5. The new build image is visible on the HERE OTA Connect server.
6. We can select the updated image from the list of committed update on HERE OTA and click to install the new update.
7. Once the update successful the target device rebooted with new image.

**Current Validation done -**

**We have build the OTA support “meta-updater” for minnowboard and tested the SoC SW upgrade over the air on Qemu and minnow board using the HERE OTA Connect.**

**Reference –**

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

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