Outstanding use cases from previous SoW:

- Multi-party call scenarios such as call waiting and placing an existing call on hold in order to place a second call.
- Play a local video from the multimedia app. Using a specific video player rather than an integrated app with audio is also acceptable. This use case is dependent on support from the AGL HMI Framework.
- Use TTS to read a text message.
- Use Speech Recognition to compose and send a text message.
- Use Speech Recognition and TTS to control the media player.
    - This particular item is potentially doable with extension of the current Alexa integration, see below

Potential new work:

- If a rework of the AGL demo is envisioned that would use landscape instead of portrait mode, rework the Qt demo applications to work in landscape mode, potentially supporting both modes via startup logic to determine orientation. Automatic scaling support has also previously been mentioned by members of the community; if desired that will increase the required effort.
- Improve demo Qt phone application UI now that HFP support is targeted for usability in Icefish. Suggested improvements:
    - Grey dialpad when HFP disconnected
    - Grey contacts, recents, and dialpad recents when PBAP not connected
    - Add phone number type icons to the contact screen and call logs screen (mobile, home, work)
    - Replace non-matching material design icons with AGL icons
    - Allow select and dialing of any phone numbers for a contact (not just the first in the list)
    - Highlight selections in lists
    - Allow alternative sorting of contacts
    - Add a favorites tab and allow selecting contacts as favorites.
    - Improve contacts List by clicking to drop down details (condense the default view to one line)
- Continue demo Qt settings application networking configuration UI improvements, including:
    - Add support for controlling priority of technologies (e.g. Ethernet or WiFi has priority for connection over one or the other when both are connected)
    - Add support for WiFi tethering (AP mode, use Ethernet as upstream connection to start)
    - Potentially investigate support for BT tethering mode
- Rework onscreen keyboard usage in demo Qt applications to use standard Qt onscreen keyboard widget, dependent on the new windowmanagement scheme enabling such.
- Update Alexa Auto SDK usage in AGL to version 2.1, released in Dec. 2019, and potentially to subsequent releases. This would include rework of the voice core and capabilities bindings to handle changes in the SDK, and any downstream

changes required for demo applications.

- Continue improving voice capabilities integration for demo usage with Alexa:
    - Integrate HVAC support, including driver/passenger separation via integration with Microchip microphone array support.
    - Integrate Alexa streaming music playback support into the existing mediaplayer binding and demo Qt application.
    - Investigate using local playback control capabilities with mediaplayer binding.
    - Improve onscreen display of weather response from Alexa to better render weather template (7 day forecast).
    - Improve onscreen display of responses from Alexa to include lists of locations template.
    - Improve integration with navigation to pass location details to navigation application via navigation API, with enhancement of the demo navigation application to render location information with a popup.
- Improve driving the IVI and cluster demos with CAN data from an external simulator to resolve observed issues with using external hardware. Initial target would be to enable using the existing simple Python script simulator on the SanCloud BBE hardware or a Raspberry Pi with appropriate CAN HAT.
- If the CARLA vehicle simulator or similar becomes freely available and desirable for use with AGL, integrating support for driving the demo with such a simulator into the upstream AGL tree. The work would likely involve updating the CAN and signal composer binding configurations. Supporting existing demo features like simulated cruise control potentially would require integration work on the external simulator if it supports such features.
- Rework of the current cluster demo to support configuring an AGL build to have the cluster applications run on the IVI platform and display on a second screen attached to it. This use case is separate from a containerization scheme. Target supported platforms would be H3ULCB with Kingfisher attached, and Raspberry Pi 4.
- After the post-Icefish upgrade to the Yocto Project "zeus" release, investigate the feasibility of enabling the upstream "etnaviv" graphics drivers for NXP/Freescale i.MX8 platforms. If the current upstream state of the FOSS graphics stack works with AGL, any required changes would be committed to the AGL tree for the Jellyfish release. Alternatively, if only the NXP Vivante driver is currently workable, integrate support to make i.MX8 targets usable with it if that is desired. The envisioned test platform is the NXP iMX8M EVKB due to straightforward availability, others can be considered.
- Investigate replacing the AGL usage of the unmaintained light mediascanner (LMS) project for hotplugged storage media scanning with something derived from one of the desktop environment mediaplayers (e.g. Totem, Rhythmbox) that have such functionality. The effort would include working with the upstream community of such a project to enable a maintained library-ification of the functionality for reuse in AGL, as opposed to simply hacking it out of the project source.

- Rework of the Bluetooth bindings to talk to a new abstraction layer that would simplify proprietary Bluetooth stack enablement. The upstream AGL integration would still use BlueZ by default via the new abstraction layer, with the abstraction layer and BlueZ back end implementation being a standalone open-source project developed by Konsulko Group.
- Work with upstream OpenEmbedded (OE) / Yocto Project (YP) to implement desired binary package feed improvements within OE / YP, with an eye on delivering improvements in the November 2020 3.2 YP release for AGL UCB 12 integration.
- Add a V4L2 radio tuner API back end to the radio binding for use with Linux kernel tuner drivers. This would also allow eventually switching to a more robust V4L2 driver for the Si4689 tuner used on the Kingfisher board that Konsulko currently has in development for the upstream Linux kernel.
- Remove baked in ForgeRock API support from AGL identity binding, replacing it with simple demo implementation based purely on local storage with the persistence binding.
- Rework the persistence binding to add global values for cross-application settings in addition to per-application persistence. This would simplify enabling demo functionality like imperial/metric units display, day/night mode, etc. configuration. Alternatively, working up a new settings binding that sits on top of the persistence binding or other backend would be feasible if desired.
- Investigate the addition of permissions to binding APIs to start the process of locking down application permissions as would be required for productization. This is somewhat dependent on the stability of the application framework and its security model.
- Support planned rework of the security model in Icefish/Jellyfish which will require changes to the start up code of all bindings and applications.
- Evaluate the effort required to switch to networkmanager and potentially modemmanager for network configuration instead of the current connman and ofono. If desired, this effort would include proceeding to attempt integration into the AGL IVI and telematics platforms and continuing on to rework the network and telephony bindings as necessary.
- Investigate integration of support for a desired Vehicle to Cloud (V2C) or Vehicle to Infrastructure (V2X) stack into the upstream AGL tree, with an eye on enabling demos of the functionality.
- Investigate the feasibility of building an open source autonomous driving stack such as Baidu Apollo (apollo.auto) or Autoware (autoware.auto) into an AGL autonomous research profile. Initially the work would consist of determining if required kernel features are available and can be enabled in AGL, and if the software stack(s) can be built using the OpenEmbedded-based AGL build. If both prove feasible, then work could proceed to add the required layers / recipes to the AGL distribution, and fleshing out the ADAS / Autonomous profile for headless operation of hardware focused on autonomous processing. Alternatively, instead of one of the two stacks previously mentioned, the autonomous profile could start with integration of the ROS stack using the

meta-ros OpenEmbedded layer, with further discussion around what a suitable demo using it should consist of.