DAB Partner User Guide

Context

The burden on Consumer Electronics OEMs to comply with OTT app testing requirements has grown as the number of OTT apps has proliferated.

Some device's partners implement custom methods to connect to devices and execute commands in their infrastructure in an automated manner. For example:

- Infrared emitter to emulate the remote control
- Mapping of UI workflow in test scripts
- Proprietary protocol to control device

The Device Automation Bus (DAB), provides a standard way to connect to devices and execute automated commands: start/stop/restart applications, perform device configuration changes, issue key presses and voice commands, collect telemetry, and more.

What is DAB?

DAB provides a standard, open-source, programmatic way to do the following with any DAB-capable device:

- Get the list of DAB-capable applications installed on the device
- Launch, suspend, stop, or kill an application
- Send short, long, and custom keystrokes to the platform
- Set/Get device system settings (i.e. networking(get), audio and video mode, tts, etc)
- Capture screenshots to automate App launch and other manual validation use cases
- Collect Device and application telemetry during testing
- Provide Voice prompts for app launch and interaction
- Get test result notifications (error, warning, info,..)

Given this diverse capability set, it is possible to develop automated test components that can be used to integrate with any of the DAB-capable application test infrastructures. You can also test interactions between multiple applications.

Why DAB?

By supporting a standardized test automation framework to be used across multiple streaming services, device partners can significantly reduce their development and maintenance costs to support diverse test infrastructures. At the same time, DAB can enable the automation of many manual tests, which reduces both subjectivity and the overall testing LOE across all the involved entities.

This reduced subjectivity, along with an ability to monitor device and application telemetry across different test scenarios, can also improve the stability of the partner device's platform.

The improvement of the partner platform quality would come from:

- Replacement of subjective manual tests with automated and objective testing
- Standard device automation interface across OTT services
- Automated application and device telemetry monitoring during tests
- Increased automated test coverage unlocked via additional test automation areas

The improvement towards testing efficiency would come from:

- Reduction of the number of manual tests
- Automated device control/setup
- Enabling OTT providers to reuse, extend, or share tests.

Now let's look at some DAB-enabled use cases.

Some Example Use Cases

Application Interaction

More streaming services are integrated into partner devices every day. This causes issues during multi-app interaction scenarios where switching in and out of apps could cause crashes or a lack of response from applications, thereby impacting the end-user experience.

As of now, many device partners only validate multi-app interactions through manual tests. By delivering an automated way to test these scenarios, DAB will make it easier for device partners to ensure that different streaming applications work well together on their devices. This could also help reduce the overall testing time and improve quality, as current manual tests can be subjective and time-consuming.

Example Scenario

Pre-condition

- The device supports DAB across multiple OTT apps.

Steps

1. Get the list of installed applications where DAB is supported.
2. Launch a DAB-compatible app A into the background.
3. Switch to a second DAB-compatible app B.
4. Exit app B.
5. Launch app A to playback.
6. Verify playback is successful.
7. Stop playback.
8. Suspend app A.

Post-condition

- All used OTT applications stopped and device resources were exchanged during app transitions.

Device Configuration

As part of streaming application testing, a partner device may need to be configured before the start of a test to ensure the relevance of the test results. This is currently a manual process that is time-consuming and would need to be repeated before each test. By automating these configuration steps, device partners and OTT app providers can reduce the time taken to run those tests and at the same time ensure that the test results are valid. Some examples of such configuration changes are Audio and Video Output settings, Network changes, System language, TTS, and more.

Example Scenario

Pre-condition

- A new device is being introduced to a lab for testing.

Steps

1. Set the audio output source via DAB to HDMI (ARC on sink device) and audio output mode to DDP 5.1.
2. Perform the AV Sync test that ensures that in DDP 5.1 audio output mode the device maintains AV Sync on its HDMI (ARC for sink) output.
3. Set the audio output mode via DAB to Stereo.
4. Repeat step 2.

Post-condition

- AV Sync across audio output formats is validated in an automated manner with DAB.

Device Remote Control and Screenshot Capture

Many streaming application tests require a tester to insert key presses to navigate to the right place in the System or Application UI. These tests are time-consuming due to their manual nature. Through DAB and with proper on-device DAB support, a device UI can be controlled by sending simulated keystrokes to the device via the streaming application partner’s test infrastructure. This, coupled with the image capture functionality on a device UI, can enable automation of numerous new test cases such as:

- Application functionality and UI navigation
    - Insert key presses to navigate different production journeys inside streaming application UIs and/or device UI.

- Application launch points

    - Automate the verification of application asset placement on the device partner platform UI.

- Inter-app switching test
    - The capability to automate the UI navigation and image capture can also be leveraged to satisfy the earlier-mentioned use case of multi-app interaction.

Example Scenario

Pre-condition

- The device has DAB enabled and is available for a connection.

Steps

1. Run a script that enables you to send keystrokes to the device.
2. The script also records the keystrokes that are sent.
3. Operate the device to execute a particular UI navigation scenario.
4. Stop the keystroke script.
5. Capture images at relevant points during the test.

Post-condition

- Submit captured images as reference.

Device and Application Telemetry

OTT providers require a certain level of device resources to enable their applications to run on a device. For example, during app integration and testing, a device partner may need to monitor the memory usage of the different applications to ensure that their integration uses the required memory. In the same vein, a device partner may also need to ensure that the CPU is not overused when the application is running, as CPU over-usage could potentially impact the behavior of the device. Finally, the partner device might also have the ability to collect streaming application-specific and device telemetry around scenarios like playback quality, which would be useful to analyze in the context of a playback test.

The telemetry endpoints provided within the DAB protocol enable easy exposure of all these metrics into the OTT testing infrastructure, thereby enabling a path for the analysis of these numbers in the context of different OTT app test cases.

Example Scenario

Pre-condition

- The device supports DAB with at least one OTT application.

Steps
1. Get the list of installed applications where DAB is supported.
2. For the application of interest:
    a. Enable telemetry reporting for the specific application and device.
    b. Launch the application.
    c. Confirm application launched.
    d. Stop the application.
    e. Launch the application to a specific title.
    f. Pause playback.
    g. Resume playback.
    h. Stop the application.
    i. Disable telemetry reporting for the specific application and device.
    j. Review reported telemetry.

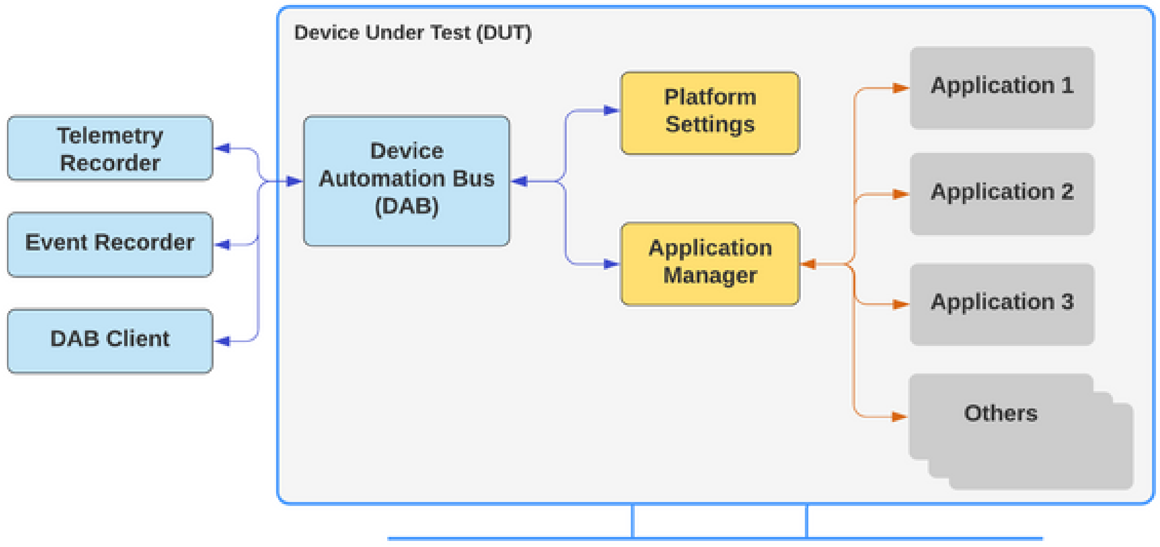
Post-condition
- Telemetry is available for review.

DAB Implementation Approach

DAB device automation can be implemented in one of the two following ways, depending on the device manufacturer's test automation framework.

Native Mode (Preferred)

The native mode requires a device partner to implement all the DAB aspects natively on the device. These are actions like creating a DAB-compatible MQTT client on the device, listening to MQTT DAB requests from the streaming app provider's test infrastructure, executing these requested DAB actions via native platform hooks, and responding through MQTT messages to report the status of that last executed action. All request/response formats should be as specified in the latest DAB specification. Given below is a conceptual diagram of different on-device and external logical components in Native mode:

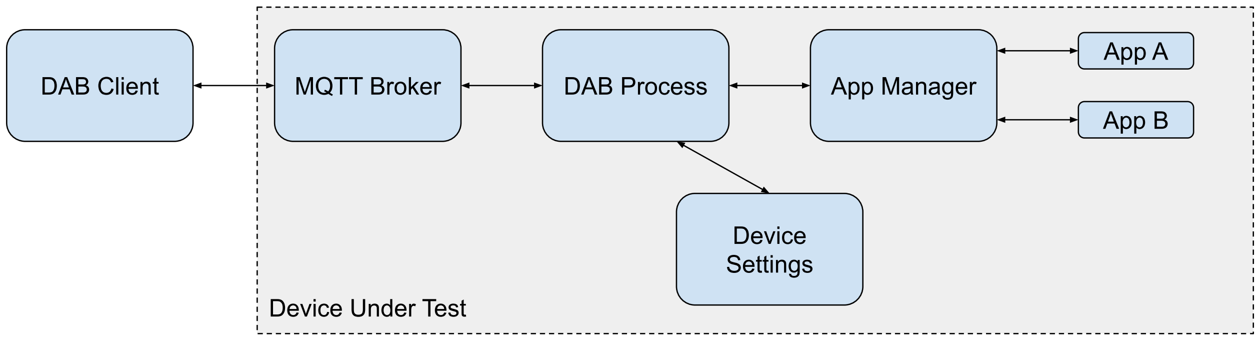


Details: In Native mode, the implementer will be responsible for deploying or implementing the elements that comprise the box labeled “Device Automation Bus” above. Those items are:

1. An MQTT broker, such as Eclipse Mosquitto. An MQTT broker is a key component of a DAB implementation.

2. A process on the device capable of connecting to the MQTT broker using the MQTT protocol. For this guide, this process will be referred to as the “DAB process.” This process will be responsible for servicing the topics and messages outlined in different sections of the DAB 2.0 specification.

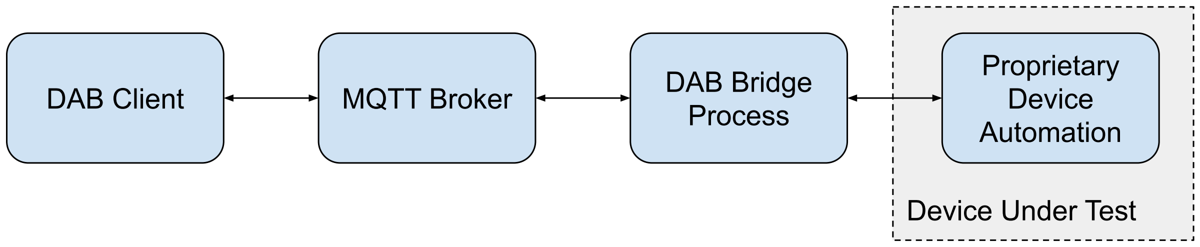
3. Native platform hooks (APIs, etc.) are accessible to the DAB process to execute different requested DAB actions on the device.



DAB Native Mode Implementation

Bridge Mode

For device platforms that already provide native automation frameworks that are accessible on the network, device partners can develop a Bridge mode DAB implementation, where the relevant DAB elements will run external to the device and interface with the device's native automation protocol.



DAB Bridge Mode Implementation

Details: In Bridge mode, the implementer will be responsible for developing the following components:

1. A DAB bridge process running external to the device that can translate the DAB request and response messages into the device-understood automation commands. This process will satisfy all the requirements mentioned in the latest DAB spec. The streaming application test infrastructure (everything to the left of the DAB Bridge Process above) will see no logical difference between a Bridge or Native mode DAB implementation.

2. A Native automation framework on the device that can perform all the automation actions specified in the latest DAB spec and interface with the DAB external bridge process.
3. The DAB Bridge implementation must handle all device varieties and maintain backward compatibility by adapting responses based on available capabilities.

Native vs. Bridge Mode

Native DAB implementation is the preferred mode of DAB support since it avoids the complexity of running and maintaining DAB process instances outside of the device platform. Native mode presents the device as a single and complete DAB-capable entity, which simplifies the test infrastructure, improves reliability, and makes debugging easier for the end-to-end DAB call flow.

DAB Implementation Requirements

Although various organizations may choose to impose additional standards for a DAB implementation, basic implementations of the DAB protocol have the following requirements:

- Your DAB implementation must either be:
    - Natively supported in your device firmware
    - -or-
    - Supported using software that translates the DAB protocol into a protocol native to your platform.
- Your DAB implementation should pass all tests in the DAB test suite, demonstrating basic compliance with the DAB specification.
- Your native DAB implementation must be accessible to a client using a standard MQTT protocol suite.
- In the native DAB implementation, the MQTT broker MUST use the default TCP/IP port 1883 to allow for uniform MQTT connection semantics across different OTT app test infrastructures.

Latency Requirements

- For DAB operations related to Application State management e.g. App Start, Stop, and State queries, the 200 response MUST only be provided after the requested operation is completed.
- For DAB operations related to key inputs, the device must complete these operations within a reasonable latency of 1 second.

- For DAB operations related to information gathering e.g. supported list of applications, system setting list, metrics gathering, device info, etc, the device must respond to these requests within a reasonable latency of 200 ms.
- For DAB operations related to image capturing, the device must complete these operations within a reasonable latency of 2 seconds.
- For DAB operations related to voice commands, the device must complete sending voice text or voice audio to the relevant endpoints within a reasonable latency of 1 second.

Relevant Pointers

1. DAB 2.0 Specification: An up-to-date version of the DAB 2.0 specification.

2. DAB Bridge SDK: Please use this Bridge SDK code to kick-start your DAB Bridge implementation if you are not using the native DAB mode.

3. DAB Compliance Test Suite: Please use this compliance test suite to validate your DAB implementation.