Infotainment Systems Product Development

AppLink Best Practices

Version 1.0

Version Date: January 5, 2013

UNCONTROLLED COPY IF PRINTED

FORD CONFIDENTIAL

Revision History

Date	Version	Created/Modified By	Notes
1/5/2013	1.00	CK	Initial version

Table of Contents

1.1 Best Practices

1.1.1 Button Management

There are a variety of options for an application to utilize when it is active on SYNC® AppLink™. After subscribing to the desired buttons, the application will be notified of button events (OnButtonEvent) and presses (OnButtonPress).

Each button press will have one of two modes: LONG or SHORT. This could be used for switching presets, saving a currently playing station or favorite to a preset, or even thumbing up or down a song.

Each button event will have one of two modes: BUTTONUP or BUTTONDOWN. For example, this could be used in conjunction with button press mode of LONG for fast forwarding through an audio stream.

For preset buttons that exist on a head unit, pre-populate those buttons with set features, for example, a user's favorite stations. If you cannot pre-populate those buttons with features, it's good to provide text-only feedback via an Alert that suggests the button is unused or a preset is not set. On certain Head Units, buttons may not exist such as presets 7-0.. You may want to check for the buttons available in the response of your RegisterAppInterface. For additional information on each Head Unit, please see the *SYNC® TDK Quick Reference and Architecture Guide* documentation.

1.1.2 Using the Display

There are two lines of text available to your application via the Show command. As a general rule, this display is used to convey the current state of your application, either what station is playing, artist/track being streamed, distance to location, or other relatively continuous updates. Lines 1 and 2.

1.1.3 Voice Recognition

1.1.3.1 Choosing Voice Recognition Commands

When designing the app, it is highly recommended to focus on voice interactions first and foremost. On a phone, navigating through an application is entirely a visual process. In a vehicle, while driving, application usage should be almost entirely accomplished using voice commands. This will help to eliminate the need to understand the driver distraction laws that are continually changing.

It may be helpful to ask the following questions:

- What are the major features/functions of my app?
  - These nouns/verbs could be loaded as top-level voice commands (*AddCommands*).
  - It is recommended that this list be less than 150 commands, to improve application initialization performance and for high voice recognition quality.
- Do I use shortcuts, favorites, presets, etc. within my app?
  - Consider automatically mapping these to the preset buttons in vehicle.
- What does a user who knows nothing about my application and has never used it before need to know?
  - A welcome message with basic instructions could be given the first few times a user uses the application on SYNC® using the Speak function. Once they've used the application a set number of times, you can remove these helpful prompts
- What does the generic 'Help' voice command do?
  - By default, SYNC® lists the application's *AddCommands* during the help prompt. The application can make the prompt more informative by changing the help prompt using the *SetGlobalProperties* command at any time.
  - Additionally, contextual help is very helpful! Often, a user can be in a certain 'section' of the application that has unique commands. Changing the help prompt to highlight those commands greatly increases user familiarity and knowledge of the application through SYNC® AppLink™.
- Is there a long list of items to select from within your application?
  - You can load these choices as items in a *ChoiceSet* and then call that *ChoiceSet* during a *PerformInteraction*.
  - Apps can load *ChoiceSet* with 100 items. If your PerformInteraction requires additional commands, you may reference additional *ChoiceSets*

- If your list of choices is known ahead of time, it is helpful to create these during your initialization phases, and simply reuse the *ChoiceSet* throughout your application's lifecycle. **Note:** It is not recommended to consistently delete and create choice sets. If you must delete a *ChoiceSet*, it is suggested that you wait some time since it was last used. Immediately deleting a *ChoiceSet* after its PerformInteraction has returned could lead to undesired application behavior

- Does my application require a logged in user to operate?
  - If your application does, you should always have logic to catch applications that are running on SYNC® AppLink™ without an active account and display a message notifying the user to log in when not driving.
- What *AddCommand*s should my audio streaming application use?
  - Audio streaming applications should consider adding the following commands at a minimum:
    - Play
    - Pause or Stop
    - Resume
    - Skip
    - Skip back, if your application allows it

Note: SYNC® GlobalSystem VR commands are not to be duplicated and will be rejected by SYNC®
Examples (but not limited to):

- USB
- Bluetooth Audio
- AM
- FM
- Phone
- Navigation
- Cancel
- Help

1.1.3.2 Effectively responding to a user with voice or display updates

It is helpful to repeat voice command input as confirmation to commands that are triggered via voice. For example, a voice command to "Get Local Traffic" would be followed up with a Speak saying "Getting Local Traffic." An exception to this practice would be while streaming audio where you may not want to interrupt audio playback with a *Speak*, so an Alert with no text-to-speech could update the display-only instead, for example to indicate that you have thumbed up or liked a song.

When designing your voice tree, be aware that you can trigger up to 3 voice prompts in a row using PerformInteraction. This is due to both driver distraction rules as well as ease-of-use. The following is an example:

- SYNC®: "Please select a country."
- User: "USA"
- SYNC®: "USA, please select a state."
- User: "Michigan"
- SYNC®: "Michigan, please select a city."
- User: "Detroit."

It is good practice to always confirm a voice command as seen in the above example where SYNC is confirming the previous response. You do NOT need to confirm the command by asking the user a yes or no question; SYNC®'s voice engine does this for you if an utterance is unclear.

1.1.4 Initializing your application

Within your initialization of code, your app is going to want to register voice recognition commands, subscribe to buttons, and set up the help prompt, among other things.
When initializing, consider performing the following actions in order to speed up processing time and ease-of-use for the user.

1. SetGlobalProperties to establish your application's help and timeout prompts
2. Send a Show command to update the display with a welcome or initialization message such as "Buffering…"
3. Subscribe to buttons required for your application
4. Register voice and menu commands using the AddCommand function.
5. Perform any other initialization, including creation of choice sets.

Note: If your application streams audio, it is best to play audio immediately, either a user's favorite station, last played content, or some other default. This may start immediately, and the display should be updated accordingly. It is best to perform this after button initialization.

1.1.5 Android™

While Android™ devices connect over Bluetooth®, this adds both additional functionality and complexity as described below.

1.1.5.1 Auto Start

To have your application show up in the Mobile Applications menu (requirement 5.1.1), you will have to implement specific functionality that monitors connections to Bluetooth® devices. Typically, this is easily done with intent listeners and a background service that acts accordingly when certain events occur (e.g. reboots or power cycles to the Bluetooth® device, Bluetooth® toggles, etc.). Below is a list of some basic use cases that you can use as starting point to see if you manage the SyncProxy correctly:

- Device power cycles
- First application run
- Bluetooth® toggling

1.1.5.2 Other Android complexities

When a Bluetooth disconnect occurs the proxy will always notify the app via an onProxyClosed notification. Unfortunately the timing for this notification is not consistent across all devices. This problem can be avoided if the app listens for an ACL_DISCONNECT and treats it as an onProxyClosed (dispose and recreate the proxy, take off the lockscreen, etc.). The Hello AppLink and AppLink Tester applications in the SDK are examples on how this can be accomplished.

1.1.6 iOS™

With iOS™ users, they expect apps to remain running, possibly continuing to streaming audio, when the application is backgrounded, and this fact is especially true when it is operating through SYNC® AppLink™. If you don't implement the following background mode, the application will be disconnected from SYNC® whenever the application is backgrounded, specifically seen with phone calls.

In the Xcode project .plist file, simply add a "UIBackgroundModes" item, which is converted to "Required background modes". Then, add at least this item, "external-accessory". Other background modes (such as audio, navigation, etc.) can be added if needed.

Also, make sure the application declares "com.ford.sync.prot0" as a supported external accessory protocol. This is required to communicate with SYNC over External Accessory.

▼Required background modes	Array	(1 item)
Item 0	String	external-accessory
▼Supported external accessory protocols	Array	(1 item)
Item 0	String	com.ford.sync.prot0

Figure 2: Sample .plist file

The application can only establish a connection with SYNC® AppLink™ while it is in the foreground. We also recommend that the phone is locked for SYNC® AppLink™ to work properly for iOS™ versions prior to iOS™ 5.0.

When two or more SYNC® AppLink™ enabled applications are present on an iOS™ device, the application that is in the foreground will maintain that connection. If a second application then connects to SYNC® AppLink™, the

first application's connection to SYNC® AppLink™ will be terminated and will have to reconnect when it is foregrounded.

1.1.6.1 Siphon Debug Tool

The Siphon debug tool is extremely useful for iOS™ application development. Because the device must be connected via USB cable to the Sync Module, the developer cannot utilize existing debugging tools, like

```
NSLog(@"Created SyncProxy.");
```

Instead, you can use the Siphon Debug Tool to produce output statements. For example:

```
[FMDebugTool logInfo:@"Created SyncProxy."];
```

See the Siphon Debug Tool guide for more information.

1.1.6.2 SyncProxy.a

The SyncProxy.a must be located in the same root directory as the project folder itself.

1.1.7 Lock Screen

As per Ford Motor Company's driver distraction rules, we require any SYNC® AppLink™ to implement a lockscreen when it is given an *HMI Level* of *FULL*, *LIMITED* or *BACKGROUND*. This lockscreen must perform the following:

- Limit application usability from the mobile device
  - Full-screen static image or view
- Show the SYNC® logo, along with your own

Additionally, you can add limited functionality to your lockscreen such as, but not limited to, the following:

- Disconnect button
  - If implemented, you should call the reset method of the SyncProxy object and then clear the lockscreen.

Note: Since iOS™ AppLink™ apps connect via USB, it is currently unnecessary to include a Disconnect button on an app's lockscreen and/or use the UnregisterAppInterface RPC. Apps will automatically disconnect when the device is ejected.

- Quick Reference Information
  - This can be a list of high level voice commands and button controls, but should not be interactive.

All additionally added lockscreen functionality will need to abide by all driver distraction rules and be approved by Ford Motor Company before the application can be submitted to its respective app store(s).

1.1.8 Command & ChoiceSet Management

While DeleteCommand and DeleteInteractionChoiceSet are supported RPCs, only use them when appropriate. Avoid deleting Commands and ChoiceSets that will knowingly be used again.