Connected Services Product Development

AppLink™ Android™ Developers Guide

SyncProxy 1.6

Version 1.6.0

Version Date: January 5, 2013

UNCONTROLLED COPY IF PRINTED

FORD CONFIDENTIAL

Revision History

Date	Version	Created/Modified By	Notes
5/22/12	0.1	CK	Initial version.
8/6/12	0.5	CK	Updated content and formatting.
8/10/12	0.5.1	CK	Added in Enum examples.
8/13/12	0.5.2	CK	Added index for RPC, Notifications, Structures, Enums. Changed document name.
8/20/12	0.5.3	CK	Added missing diagrams. Added new items to iOS FAQ.
10/2/12	0.5.3.1	CK	Added new requirements 5.1.11, 5.1.12.
10/10/12	0.5.3.2	CK	Removed extra language in 3.3. Updated reference documents section 1.4.
12/19/12	0.5.4	EW	Updated to break out iOS and Android into separate guides. Updated formatting of various sections. Added Section 6.3.
12/21/12	0.5.5	EW	Updated to include changes for SyncProxy 1.6.
1/4/13	1.6.0	CK	Changed version to 1.6.0 to match SyncProxy 1.6.

Table of Contents

1 Introduction

1.1 Disclaimer, Copyrights, and Trademarks Notice

SYNC® and AppLink™ are trademarks of Ford Motor Company. All other trademarks are the property of their respective owners.

1.2 Purpose

The purpose of this document is to give various user experience descriptions of how SYNC® with Mobile Application Connectivity will operate from the perspective of a driver in the vehicle. Technical descriptions, functions and capabilities identified in this document are intended to serve as examples only, and do not necessarily intend to provide guidance towards an engineering solution. Within this document, examples will be highlighted as such, and should not constitute a requirement unless explicitly identified as such.

1.3 Nomenclature

Term	Description
US	United States
HMI	Human Machine Interface
TTS	Text To Speech
FMC	Ford Motor Company
PTT	Push To Talk
TBT	Turn-By-Turn Navigation
VR	Voice Recognition
Mobile Gateway	SYNC® software developed under this document
Endpoint	A transport concept which identifies what a transport connects to
Connection	A transport concept
RPC	Remote Procedure Call
SYNC®	Host platform of the mobile gateway software
DID	Data Identifier
DTC	Diagnostic Trouble Code
ECU	Electronic Control Unit
Mobile Application Session	The time when a mobile application is interacting with the user

1.4 References

This section contains references to documents which affect the requirements presented in this requirement specification.

Reference Title
AppLink™ Test Cases
SYNC® TDK Quick Reference and Architecture Guide

2 Mobile Applications Definition

Mobile applications are defined as a software executable that resides on a mobile device. The Mobile Applications Menu will contain the list of all the mobile applications that are AppLink™ -enabled currently running on a paired and connected mobile device.

2.1 AppLink™ -enabled Application Definition

AppLink™ -enabled applications shall communicate with SYNC® over a known transport layer, exchanging messages in a pre-determined format. These messages include (but are not limited to) command and control information, as well as other program data, executing and utilizing existing SYNC® features to interact with the driver. Any reference to a mobile application or mobile application herein implies an AppLink™ -enabled application, indicating one that is modified to send the proper messages to SYNC®.

3 AppLink™ Definitions

3.1 What is SYNC®?

SYNC® is an integrated in-vehicle communications and entertainment system that allows customers to make and receive hands-free telephone calls, control music, have turn-by-turn directions, and more simply by utilizing your voice. In January 2012, SYNC® has been installed in over four million vehicles and is projected to top nine million vehicles by 2015.

3.2 What is SYNC® AppLink™?

SYNC® AppLink™ is intended to provide the capability for third party developers of mobile device applications to AppLink™ -enable their products. Once this mobile gateway is completed, mobile applications will have the ability to present their HMI to the vehicle occupants in a safe, non-distracting and consistent way. The applications are intended to run on the mobile device, and exchange program data as well as command and control information through the SYNC® gateway. This technology is similar to how Bluetooth phones and digital media are integrated and used on the current SYNC® production platform.

3.2.1 Communication Mechanisms

Android™ applications communicate with SYNC® AppLink™ over Bluetooth®.

As technology may change in future versions of AppLink™, all communication will be handled by the AppLink™ APIs.

3.3 What is the SYNC® AppLink™ Proxy?

The SYNC® AppLink™ proxy, or SyncProxy, is the intermediary that sits between your app and the vehicle and marshals information to and from. The SyncProxy directly interfaces with SYNC®'s in-vehicle Human Machine Interface (HMI) and uses Remote Procedure Calls (RPCs) and callbacks for communication (e.g. display text, speak a phrase, start audio capture, button-pushes, completion of asynchronous operations, etc.).

3.3.1 Automated Lifecycle Management (ALM)

When using the Automated Lifecycle Management, or ALM, version of the SyncProxy, SyncProxyALM, the SyncProxyALM manages the lifecycle of the connection to SYNC® on behalf of the application. Once the SyncProxyALM object has been instantiated, the application simply needs to monitor and respond to lifecycle callbacks on the IProxyListenerALM interface.

It is very important to monitor and respond to the lifecycle events in the IProxyListenerALM. Failure to do so will result in undesired behavior. This will not only affect the functionality of your application, but will fail the validation tests required to have your application approved.

The entire lifecycle can be summarized by the figure below:

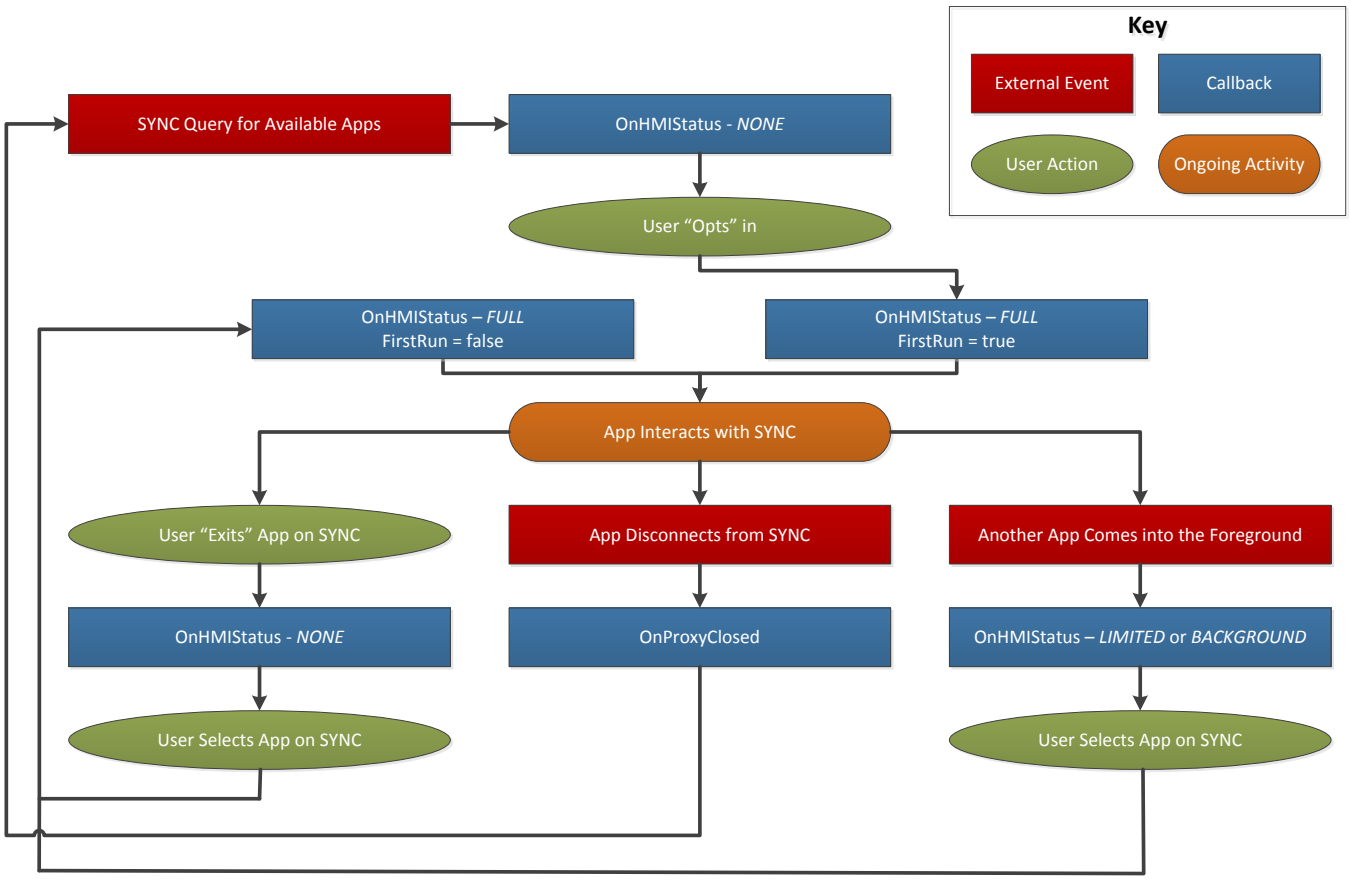

Figure 1: SyncProxyALM Lifecycle

The SyncProxyALM object only needs to be disposed of and discarded when the application no longer requires further communication with SYNC® or an unrecoverable error occurs. Otherwise, the SyncProxyALM object will manage the connections to and subsequent disconnections from any AppLink™ enabled SYNC® units for the duration of its lifecycle.

3.3.1.1 Connecting SyncProxyALM

The SyncProxyALM object exposes several constructors for instantiation. Through these constructors, the application can pass in important information, such as the application's name, which the SyncProxyALM will use to register an interface with SYNC® on behalf of the application.

When building a SyncProxyALM instance through the most basic constructor, the application must provide an implementation of the *IProxyListenerALM*, its name, and a Boolean to indicate whether or not the application is a media application, and an implementation of the *IProxyListenerALM* interface. This *IProxyListenerALM* interface, the app's proxy listener, will be the object that receives callbacks when SYNC® sends responses and notifications to the app.

Building a SyncProxy instance enables the application to be found by SYNC®, but does not mean that the application is immediately connected. SYNC® drives the initial connection to the device in response to user interaction in the vehicle. The application will know SYNC® is connected when it receives an onHMIStatus callback on its proxy listener.

When the application receives an onHMIStatus notification with an HMI level of FULL, the application may begin sending requests to SYNC®.

3.3.1.2 Disconnecting SyncProxyALM

There are three ways a registered application can disconnect from SYNC®:

- The application requests that the proxy disconnects from SYNC®.

- The application calls the resetProxy method. When this method is called, the SyncProxyALM disconnects from any existing connection with SYNC®, disposes of all transient resources and reinitializes itself. Once the initialization has completed, the proxy will be able to accept new connections from SYNC®.
- The application calls the dispose method. When this method is called, the SyncProxyALM permanently disconnects form any existing connection with SYNC®, disposes of all resources. At this point, the SyncProxyALM object can be discarded. A new SyncProxyALM object will need to be instantiated before further communication with SYNC® can occur.

- If the connection between SYNC® and the application is expectedly or unexpectedly lost (e.g. the mobile device and vehicle are out of range of each other, etc.), the application will be notified via the OnProxyClosed callback receiving SYNC_PROXY_CYCLED exception. The SyncProxyALM will continue to monitor for new SYNC® connections.
- If an unrecoverable error, which cannot be resolved without intervention by the application, occurs in the proxy, the application will be notified via the OnProxyClosed callback. At this point, the SyncProxyALM object must be disposed of and discarded. A new SyncProxyALM object will need to be instantiated before further communication with SYNC® can occur.

3.3.1.3 Multiple SyncProxyALM Instances

An application cannot instantiate more than one concurrent SyncProxyALM objects, though SyncProxyALM objects can be instantiated serially. This will make the application unusable by the user.

3.4 What is SYNC® HMI?

SYNC® HMI is a term to describe the interface that the user/driver uses to interact with the vehicle. This interface includes a collection of presets, media buttons (seek forward/backward, tune up/down, and play/pause), menu items, and most importantly voice commands. Please note the buttons available will vary between vehicles/head units.

For more details about the available buttons per head unit type, please see the *Head Unit Architecture Reference* documentation.

3.4.1 Important HMI Lifecycle Concepts

Important HMI Lifecycle concepts include:

- Command Menu (user-initiated interaction)
- Interaction Mode (Voice Recognition (VR) versus Menu interactions)
- SYNC® Connection State
- HMIStatus
- HMILevel
- AudioStreamingState
- SystemContext Change

3.4.1.1 Command Menu

This is the list of available commands, available through the menu or VR, to the user. It is composed of submenus and commands.

3.4.1.2 Interaction Mode

This is specific to a given command and denotes how it can be accessed by a user. Commands in the menu can be accessed through the menu operations available in the HMI, VR, or both.

3.4.1.3 SYNC® Connection State

Indicates whether or not the application has any connection to a SYNC® module. This state is indicated to the application through callbacks on the IProxyListener interface.

3.4.1.4 HMIStatus

This is an important RPC Message that signals the connected application what rights SYNC® is currently granting it.

3.4.1.5 HMI Status Level

Indicates the level of access to the SYNC® HMI which is currently granted to the application through the proxy. This state is indicated to the application through the OnHMIStatus callback on the IProxyListener interface.

3.4.1.6 AudioStreamingState

Indicates whether or not an audio streamed by an application will be audible to the user. This state is indicated to the application through the OnHMIStatus callback on the IProxyListener interface.

3.4.1.7 SystemContext

Indicates whether or not a user-initiated interaction is in progress, and if so, in what mode (i.e. MENU or VR). This state is indicated to the application through the OnHMIStatus callback on the IProxyListener interface.

3.4.2 Role of Connection State

In the IProxyListener interface, the OnProxyOpened (for Legacy) and the OnProxyClosed notifications convey to the application whether or not the application has a connection to a SYNC® module.

3.4.2.1 OnProxyOpened (for Legacy)

Once the OnProxyOpened notification is received the application is now connected to SYNC® and must send a RegisterAppInterface in order to finish the connection and register in the mobile apps menu. This process is complete and your app will appear in the mobile apps menu once it receives an HMI Level of HMI_NONE

3.4.2.2 OnProxyClosed

Once the OnProxyClosed notification is received the application no longer has an active connection to SYNC®. Any menu commands, choice sets or button subscriptions have been deleted on SYNC® and must be recreated upon the next connection to SYNC®.

3.4.3 Role of HMI Status

While the *RPC Reference documentation* will enumerate all RPCs, notifications, and their parameters, there is one callback that deserves special mention. The *OnHMIStatus* RPC notification conveys information regarding the state of the HMI.

OnHMIStatus declares an application's HMI level (whether the application is actively displaying information to the driver) and audio streaming state (whether the application can stream audio to the driver). This information is used by the application to determine which RPCs may be called.

3.4.4 HMI Status Levels

While the SYNC® AppLink™ application model permits multiple applications to be concurrently active and connected to SYNC®, only one of those applications may communicate at a time with the user using the SYNC® HMI.

AppLink™ uses the concept of *HMI Levels* to describe the current state of the application with regards to the level at which SYNC® can communicate with it (and vice versa).

HMI Levels are represented in four states:

- FULL
- LIMITED
- BACKGROUND
- NONE

3.4.4.1 FULL

Once the HMI_FULL level is received by the application, the application has full access to the SYNC® in-vehicle user interface. This occurs most often when a user "opts" in to using this application, either via voice or selecting the application from the Mobile Applications menu. Applications will also return to this state when when SYNC® ends a system process and turns its attention back on the mobile application.

3.4.4.2 LIMITED

Once the HMI_LIMITED level is received by the application, the application only has limited access to the SYNC® in-vehicle user interface. This *Focus Level* only applies to navigation-equipped SYNC® units (NGN).

3.4.4.3 BACKGROUND

Once the HMI_BACKGROUND level is received by the application, the application no longer has focus on SYNC®. In this level, SYNC® sees the current application as a secondary process (e.g. due to another SYNC® application opening or a user-initiated event) and as such, the user cannot directly interact with the application (e.g. Push-to-talk (PTT), menu commands, etc.) and the application cannot directly interact with the user (e.g. *PerformInteraction*, *Alert*, etc.).

3.4.4.4 NONE

In this level, the application has registered its interface, but cannot do anything.

Any operation which is started (but not yet terminated) by an application while that application has an HMI Level of *FULL* or *LIMITED*, will automatically be terminated when that application loses focus on SYNC®, *BACKGROUND* or *NONE* (e.g. interactions in-progress, *Speaks* in-progress, etc.)

3.4.5 The Role of Audio Streaming State

In the IProxyListener interface, the OnHMIStatus notification informs the application that there has been a change to their ability to stream audio over the vehicle's audio bus.

3.4.5.1 Audio Streaming State: AUDIBLE

Once the notification OnHMIStatus is received with an AudioStreamingState value "AUDIBLE", currently streaming audio, if any, is audible to the user through SYNC®.

3.4.5.2 Audio Streaming State: NOT_AUDIBLE

Once the notification OnHMIStatus is received with an AudioStreamingState value "NOT_AUDIBLE", currently streaming audio, if any, is not audible to the user through SYNC®.

3.4.6 The Role of SystemContext

In the IProxyListener interface, the OnHMIStatus notification informs the application whether or not a user-initiated interaction is in progress, and if so, in what mode (i.e. MENU or VR).

3.4.6.1 SystemContext: MAIN

When the notification OnHMIStatus is received with a SystemContext value "MAIN", No user interaction (user-initiated or app-initiated) is in progress.

3.4.6.2 SystemContext: VRSESSION

When the notification OnHMIStatus is received with a SystemContext value "VRSESSION", VR-oriented (user-initiated or app-initiated) interaction is in-progress.

3.4.6.3 SystemContext: MENU

When the notification OnHMIStatus is received with a SystemContext value "MENU", Menu-oriented (user-initiated or app-initiated) interaction is in-progress.

4 General Practices and Frequently Asked Questions

4.1 Best Practices

4.1.1 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.

4.1.2 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.

4.1.3 Voice Recognition

4.1.3.1 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 *ChoiceSet*s
  - 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

4.1.3.2 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.

4.1.4 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.

4.1.5 Android™

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

4.1.5.1 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

4.1.5.2 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.

4.1.6 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.
- 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).

4.1.7 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.

4.2 Frequently Asked Questions (FAQ)

4.2.1 General

4.2.1.1 How does "Mobile Applications -> Find New Applications" work?

When "Find New Apps" is selected the following occurs:

6. SYNC® does an SDP inquiry against the current HFP-connected device looking for endpoints with a specific service class UUID.
7. SYNC® opens RFCOMM connections to those discovered endpoints.
8. If the application accepts a connection, it is expected to call *RegisterAppInterface* within 20 seconds
9. Apps that are successfully registered will appear in the Mobile Applications menu.

Given that, SYNC® will fail to find an application for the following reasons:

- The application does not have an active SDP record with specific UUID tied to AppLink™.
- The application has the aforementioned SDP record, but is not accepting connections.
- The application has the SDP record and accepts the connection, but does not successfully register its interface.

Note: When an instance of the SyncProxy is created, it will set up the appropriate SDP record and will automatically accept **ONE** (and only one) connection. When this connection is accepted the application will be notified via *onProxyOpened* method in *IProxyListener*, after which the application must register its interface within 20 seconds.

Calling the *dispose* method will tear down the connection and SDP record. Even if the proxy listener has received the *onProxyClosed* callback, it is necessary to call the *dispose* or *close method*.

4.2.1.2 Can more than one device have active AppLink connections to SYNC® at the same time?

No. Only the device with an active Hands-Free Profile (HFP) connection to SYNC® can have active AppLink connections. SYNC® only permits one device at a time to have an active HFP connection.

There are situations in SYNC® where the user can choose an Advanced Audio Distribution Profile (A2DP) source other than the active HFP connected device, but that does not change the active HFP connection.

4.2.1.3 What should happen when a user quits the application on the mobile device?

The application should know it is being shut down and issue a SyncProxy reset so it can be rediscovered in the vehicle on the next ignition cycle of via the Find New Apps option on SYNC. Calling the reset method will prevent SYNC® from verbally reporting "...a problem in stopped communication with..." your application.

4.2.1.4 Is it possible to read out the global Media Settings menu via AppLink where one can turn on and off settings like Shuffle, etc.?

At this time you cannot read the system or media settings via SYNC® AppLink™. Apps don't have access to any form of persistent information currently, neither system nor application specific.

4.2.1.5 What is the relationship between AVRCP and AppLink apps?

When the application does not have a *HMI Level* of FULL, Audio/Video Remote Control Profile (AVRCP) events will not be sent from SYNC®. The *SubscribeButton* mechanism replaces this (and provides additional information about button activity). Also, it is possible to receive AVRCP commands. For example, if the user is in your application and pushes the AUX button, then AVRCP commands will be received through the Bluetooth® link.

4.2.1.6 When should I pause/resume audio when connected to SYNC?

See question below "How Does HMI Status Work?"

4.2.1.7 Are there any times the application CANNOT customize SYNC's default prompts?

The application cannot override the initial prompt that occurs when the user presses the PTT button. This prompt will always be "<app_name>: Please say a command". This cannot be changed because it is a system pattern that brings consistency to the SYNC® user experience. Most other in-app prompts can be customized as needed.

4.2.1.8 What VR commands should a media playing application implement?

There isn't a canonical list of VR commands, but the most prevalent convention which has come into use is to preface all the individual station names with a VR command "Play Station ...." in front of it. For example, "Play Station Cat Stevens" or "Play Station Jazz" or "Play Station My Favorites" or "Play Station Detroit Michigan" or "Play Top Ten"

Beyond this sort of "station identification" it pretty much boils down to a 1:1 mapping of what the user can do while the media stream is underway, so you might introduce VR commands like "Skip", "Hate it", "Love it", "Thumbs up", "Thumbs down" etc.

You might also have some list based activities, like "List my stations", "List my favorites".

We highly recommend that apps have ways for users to discover and bookmark new media sources, not just play the canned lists previously setup before drive time. "Bookmark song" and "find similar" might be commands that could help.

Look at existing SYNC® AppLink™ partner apps from the App Store, Marketplace, or App World for more ideas. A list of these apps can be found on the www.SyncMyRide.com site.

4.2.1.9 What is the startup delay time for SYNC® to be ready to accept a connection when the head unit is powered on?

After the ignition is turned on, SYNC® should be ready to connect within 4-5 seconds. If the battery is disconnected and then reconnected, SYNC® can take as long as 30 seconds to be ready.

4.2.2 RPC Related

4.2.2.1 What is the significance of SUCCESS in responses?

SYNC distinguishes 3 different states for responses:

- success=true, errorcode = SUCCESS
  - The request was processed without any issues. This is regarded as FULL SUCCESS.
- success=true, errocode != SUCCESS
  - The request was processed, but some (minor) issues occurred (One of the issues is reflected in the errorcode). This is regarded as a WARNING.
- success=false, (errorcode will never be SUCCESS)
  - The request couldn't have been processed because a non-recoverable error occurred. This is regarded as an ERROR.

4.2.2.2 Is incrementing the clock handled automatically by the timer?

Yes, updates occur once per second in increments of one second. These update modes support elapsed time and remaining time use cases. You may want to consider calling *SetMediaClockTimer* regularly with an updated startTime.

4.2.2.3 How does updateMode PAUSE | RESUME work?

When you call *SetMediaClockTimer* you can provide a startTime and specify that it should COUNTUP or COUNTDOWN, or you can provide no startTime and specify that it should PAUSE or RESUME counting at the current value.

4.2.2.4 Is there a limit to how fast I can send *AddCommand* requests?

If you need to send a large number of requests to set up your application up when leaving *HMI Level* of NONE, the following is recommended:

- Send a small number of the most important requests immediately. These usually include button subscriptions, custom prompts (*SetGlobalProperties*), and four or five of the most important commands.
- Using a timer or background thread, send the remaining *AddCommand* requests in batches of 5 or 10 every second or so.
- The most *AddCommand*s any application has sent at startup is around 200. This will take about 10-15 seconds. Commands sent first will be available as voice commands first. So, it's recommended to send more important commands first. It is technically possible to send more than 200 commands, but this will increase loading time for the application and decrease SYNC®'s response time to user input.

SYNC® will disconnect on these conditions:

- More than 200 RPC requests in 2 seconds.
- Mobile application sends more than 5 requests within 5 seconds in the *HMI Level* of NONE.

Since requests (while they are transmitting / receiving responses asynchronously) block the thread of execution, the SyncProxy should be treated like any other high-latency network connection.

As such blocking the UI thread with the SyncProxy commands may create delays which invoke the non-responsive termination rule on the mobile platform.

4.2.2.5 Are the two levels available in the menu display both dynamic?

Yes. You can add commands and submenus to the top-level menu. You can add or delete commands to any of the submenu. There are two entries pre-populated into the top-level menu that cannot be removed: "Exit <app_name>" and "Return="" ("Return="" brings you back to the previous menu).

4.2.2.6 Can SYNC® display 150 list items and allow user to "scroll" through them with the tuner?

This is not recommended. For functions with long lists, it is recommended to create the list of choices using *CreateInteractionChoiceSet* and trigger the interaction from the application by calling *PerformInteraction*.

An alternative instead of using the menu is to subscribe to the TUNEUP and TUNEDOWN buttons and send a *Show* command to manage the list from this event handler.

4.2.2.7 Can SYNC® call out the first letters of the item list as the user "scrolls" through them?

No. This feature is not available in current versions of SYNC® AppLink™ nor will it be in the foreseeable future. You are not the first developer to request this, so we will add it to our list of potential future features

4.2.2.8 Can we have freestyle VR? For example, I want to register for "Play station [anything]", and have the transcription of the [anything] value passed to me for handling...like the way voice command on Android works.

No. SYNC®'s VR is performed locally and for performance reasons works with predefined grammars only. The functionality you describe on Android/iOS 5 with Siri is made possible because the VR processing happens in a data center. In general, transcription is not supported through the voice engine.

4.2.2.9 When does *onProxyClosed* get called?

There are two types of disconnection resulting in *onProxyClosed*:

- Requested disconnection: through *UnregisterAppInterface*
- Forced disconnection: usually from communication failure, protocol error or SYNC® rule violations (e.g. more than 200 requests in 2 seconds, or more than 5 requests in 5 seconds during the *HMI Level* of NONE)

4.2.2.10 I'm seeing *onProxyOpened* called twice. My app's registration is being rejected because of a duplicate name. What's going on?

You may have instantiated two instances of the SyncProxy connected to the same listener, or you have the listener registered twice with the SyncProxy. If you have two instances, SYNC® connects to the endpoint advertised by each SyncProxy object, and sets up two connections.

The first registration will succeed, and the second registration will fail because it uses the same application name as the first. You will find it very difficult to manage the app's state if you have two connected proxies sending triggering callbacks on the same listener.

Make sure you only create one instance of the SyncProxy at a time. Dispose of it properly before creating another instance.

4.2.2.11 Why was my *Alert* rejected?

An *Alert* will be rejected if it is requested while another *Alert* is in progress. *Speak*s may be interrupted by *Speak* or *Alert* requests, but *Alerts* will not be interrupted for *Speak* or *Alert* requests.

4.2.2.12 Why do I receive the response info string "A mandatory label in the json message is missing: .type..."?

You may be creating your *TTSChunks* improperly. Each *TTSChunk* object must contain a "text" and "type" parameter. You might be currently creating the *TTSChunk* like this:

- TTSChunk chunk = new TTSChunk(); chunk.setText("custom help prompt");

You should be creating it like this:

- TTSChunk chunk1 = TTSChunkFactory.createChunk(SpeechCapabilities.TEXT, "custom help prompt");

Or you could create the collection of chunks all at once:

- Vector chunk2 = TTSChunkFactory.createSimpleTTSChunks("custom help prompt");

4.2.3 HMI Related

4.2.3.1 What happens to an application when the user selects Exit (or Quit) on that application's menu in SYNC®?

SYNC® will issue a HMI Level of NONE.

4.2.3.2 How does HMI Status work?

OnHMIStatus serves as a single notification that carries three largely orthogonal (though not strictly so) pieces of information describing what the application can do with, or expect from, the HMI. Although these three pieces of information often change independently of each other, the reason they are lumped together into a single notification is to minimize the number of RPC messages (because two or more elements often change at the same time).

- *HMI Level* describes the degree of user interaction possible. The application should watch *HMI Level* when interacting with the user via TTS, VR, Menu-Selection, Display or expecting the receipt of Button push notifications. Apps with *HMI Level* of FULL or LIMITED are generally considered to be "in focus". By "in focus", we mean that the application can receive user input and can communicate to the user through the display, TTS, *PerformInteraction*, and audio streaming.
- *AudioStreamingState* indicates both:
  - The application has (does not have) permission to stream audio
  - The application's streamed audio is audible (inaudible) to the user

Remember that SYNC® does not distinguish between which apps are streaming on the device. As a result, this indicates permission, and your application must not play audio to prevent blending audio streams.

As an informational message (e.g. a phone call is answered and the application's audio is no longer audible), this indicates that the application's audio stream is inaudible (regardless of whether it is being played by the application or not).

- SystemContext
  - MAIN: no interaction in progress
  - VRSESSION: VR is active as a result of PTT or *PerformInteraction*
  - MENU: a menu interaction is in progress

The application should watch *SystemContext* when a user-initiated interaction is in progress (e.g. pressing the PTT or menu button).

These are not strictly orthogonal because there are certainly some relationships among them (e.g. a SystemContext of VRSESSION will always correspond to AudioStreamingState of NOT_AUDIBLE, etc.).

But each of the three elements of HMIStatus should be examined independently of the others, depending on what thing(s) the application needs to do with the HMI.

4.2.3.3 What is the difference between *HMI Levels* NONE and BACKGROUND?

NONE means it is known about by SYNC® but cannot issue or receive RPC calls except for *UnregisterAppInterface*.

BACKGROUND means that a limited set of RPC calls can be issued or received.

NONE is sent immediately after successful application interface registration when SYNC® establishes a connection with the app. When in NONE, the application is available from the Mobile Applications menu but may not issue any requests (i.e. consume system resources) because the user doesn't know about it unless they inspect the Mobile Applications menu.

When the user selects an application from the Mobile Applications menu, we consider the user to have opted-in and grant that application access to system resources (meaning requests may now be issued). An application that has been opted-in will always have some *HMI Level*.

The application will be returned to NONE when the user selects "Exit <app_name>" via the menu or PTT VR command. The user has opted out of using the application at this point and the application may not send requests. The application's registration, button subscriptions, display state, custom prompts, interaction *ChoiceSets*, and commands will be persisted and available if the user again selects the application from the Mobile Applications menu.

4.2.3.4 What is the expected initialization procedure with respect to HMI Status to set up menus and commands?

The expected procedure with respect to HMIStatus for initialization is that when the application first receives a *HMI Level* other than NONE, it can and should begin creating resources, such as commands/menus through *AddCommands*, *ChoiceSets*, and button subscriptions. The application should begin creating resources at this time because the user could expect interaction to be possible at any time through PTT or menus (user initiated).

4.2.4 Android

4.2.4.1 How does the SYNC® HU trigger an ACTION_ACL_CONNECTED intent?

ACL_CONNECTED and ACL_DISCONNECTED are seen when BT connections to the Android device are established or destroyed. The most common time this is seen is when HFP is connected (i.e. with the [phone icon] button) pressed or disconnected (i.e. switch the KEY OFF and wait 30-60 seconds if needed.)

4.2.4.2 Can an application that was connected auto re-connect/register to the SyncProxy? It seems that I have to go through the "find SYNC apps" process on the head unit every time I shut the application down even though the device remains connected.

To make sure that your application appears promptly in the Mobile Applications menu, it is recommended to instantiate the SyncProxy during startup (i.e. before the phone connects to SYNC® via HFP). Mobile apps are automatically discovered when the HFP connection is established.

Many current apps have accomplished this by instantiating the SyncProxy in the background on device boot. For a demonstration of this, see the sample app, *SyncProxyTester*.

5 AppLink™ Requirements

5.1 General

5.1.1 Your app shall always be available in the mobile apps menu when the phone is connected[1] to the vehicle.

5.1.2 When application is running through AppLink™ and an incoming call is received, the audio channel shall be paused while your app is in BACKGROUND / NOT_AUDIBLE.

5.1.3 When application is running through AppLink™ and an outgoing call is placed, the audio channel shall be paused while your app is in BACKGROUND / NOT_AUDIBLE.

5.1.4 Your application should pause or stop audio while in the BACKGROUND or NONE state.

5.1.5 Your application should minimally mute, preferably pause or stop when transitioned to the NON_AUDIBLE (FULL or LIMITED) state.

5.1.6 Your application shall send a *SetGlobalProperties* to establish an advanced help prompt before sending any voice commands.

5.1.7 When application transitions into the FULL, LIMITED or BACKGROUND states OR you receive an OnDriverDistraction notification DD_ON, the application shall allow no interaction with the application's UI on the mobile device.

5.1.8 When application is in NONE state, has been disconnected from SYNC or receives an OnDriverDistraction notification DD_OFF, the application can allow regular interaction with the mobile device.

5.1.9 If using the term Ford SYNC® or AppLink in your application GUI, you must include a registered symbol and trademark in the following fashions. You must ask for permission to use these trademarks or logos.

Ford SYNC®
SYNC® AppLink™

5.1.10 Applications shall not register voice grammars using synonyms that include other application names, or conflict with on-board voice commands

5.1.11 Applications shall provide an audio or video response to user interaction (button presses, VR, add-commands, etc.)

5.1.12 Applications must not provide an incorrect or unexpected response to user interaction.

5.2 Driver Distraction

Driver distraction is a very serious concern when developing an application for use in-vehicle. Rules and laws are always in flux and may change in the future, so it is important to consider these ramifications at the beginning of the development cycle.

When the SYNC® system is in "driver distraction" mode, the user will not be able to access certain areas of the HMI. For example, when the vehicle is in motion, phone pairing and sub level menus are not accessible. Designing a menu tree that is quick, intuitive, and easy to use is critical.

To help with relaying information about when the vehicle motions, the *OnDriverDistraction* RPC message has been added. This allows the application to receive notifications when the vehicle begins to move over 5 Km/h (about 3 mph) and when the vehicle comes to stop.

6 AppLink™ RPC Operations

An AppLink RPC operation is a request/response pair that instructs AppLink to perform some HMI operation, such as displaying text, speaking text, or to define other elements used in HMI communication, such as commands.

Requests are made by the application. All requests require that the application provide a correlationID parameter. All responses include a correlationID parameter and a response data structure parameter (the response data structure contains the "success," "resultCode," and "info" data elements).

A given correlationID value cannot be used in two or more concurrently active requests. That is, if a given correlationID value is provided in one operation request (e.g. *Speak*), then the same correlationID value cannot be used in another operation request until the first operation request completes (with its response).

If SYNC does not recognize an RPC request (i.e. it is not one of the RPC operations described in this section), SYNC will send a GenericResponse RPC response. The GenericResponse RPC response contains the same parameters that are in any other RPC response (success, resultCode, info, correlationID).

A correlation ID is included in a response so that it can be matched with the request which precipitated it. That correlation ID is returned to the application as a parameter in the corresponding response. It is the responsibility of the application to ensure that the correlation IDs are unique, so that a given response can be unambiguously associated with its precipitating request, if the application requires it.

The HMI Status Requirements section for each operation indicates which HMI Status conditions must hold true for valid request of the operation.

All responses contain the following three parameters (and possibly additional parameters, depending on the particular operation -- e.g. the response to the *RegisterAppInterface* request contains several additional parameters):

Parameter List

Name	Type	Description	Req.	Notes
success	Boolean	A Boolean indicating whether the original request was successfully processed.	Y	
resultCode	Result	The result code provides additional information about a response returning a failed outcome. Any response can have at least one, or possibly more, of the following result code values: SUCCESS, INVALID_DATA, OUT_OF_MEMORY, TOO_MANY_PENDING_REQUESTS, APPLICATION_NOT_REGISTERED, GENERIC_ERROR,REJECTED. Any additional result codes for a given operation will be highlighted in the corresponding section below.	Y	
info	String	A string of text representing additional information returned from SYNC. This could be useful in debugging.	Y	
correlationID	Int32	An integer value correlating the response to the corresponding request.	Y	

Note: Included in the SDK is an application called AppLinkTester, which has an example of using each RPC.

6.1 SYNC® Requirements for RPC

SYNC® will take in requests from an application at any given time, but can only process them based on the current *Connection State* and *HMI Level* of an application. For instance, if the application wants to send an *Alert* request, the application needs to have an *HMI Level* of *Full* in order for SYNC® to successfully process and execute the request. The chart below can be used as a quick reference guide for the *HMI Level* requirements of RPC Requests.

RPC Request Availability Matrix						
SYNC Connection State	**State**	**Connected**				**Not Connected**
	Callback	OnHMIStatus				OnProxyClosed
HMI Level		FULL	LIMITED	BACKGROUND	NONE	N/A
RPC Request		AddCommand				
		AddSubMenu				
		Alert				
		CreateInteractionChoiceSet				
		DeleteCommand				
		DeleteInteractionChoiceSet				
		DeleteSubMenu				
		EncodedSyncPData				
		PerformInteraction				
		ResetGlobalProperties				
		SetGlobalProperties				
		SetMediaClockTimer				
		Show*				
		Speak				
		SubscribeButton				
		UnsubscribeButton				

Notes
* show() can be performed whenever the Sync Interface is available, however this results won't be visible to the user until the application has an HMI level of either *FULL* or *LIMITED*

Figure 2: RPC Request Availability Matrix

Please see RPC sections for comprehensive details about RPC messages.

6.2 Typical RPC Exchange

The following diagram demonstrates a typical sequence of RPC exchanges for a simple application. Here, an application registers the application interface with SYNC®, populates a menu with a command, and speaks a welcome message. The diagram assumes that a transport connection and a transmission protocol RPC session have been established, and that RPCs are being marshaled appropriately.

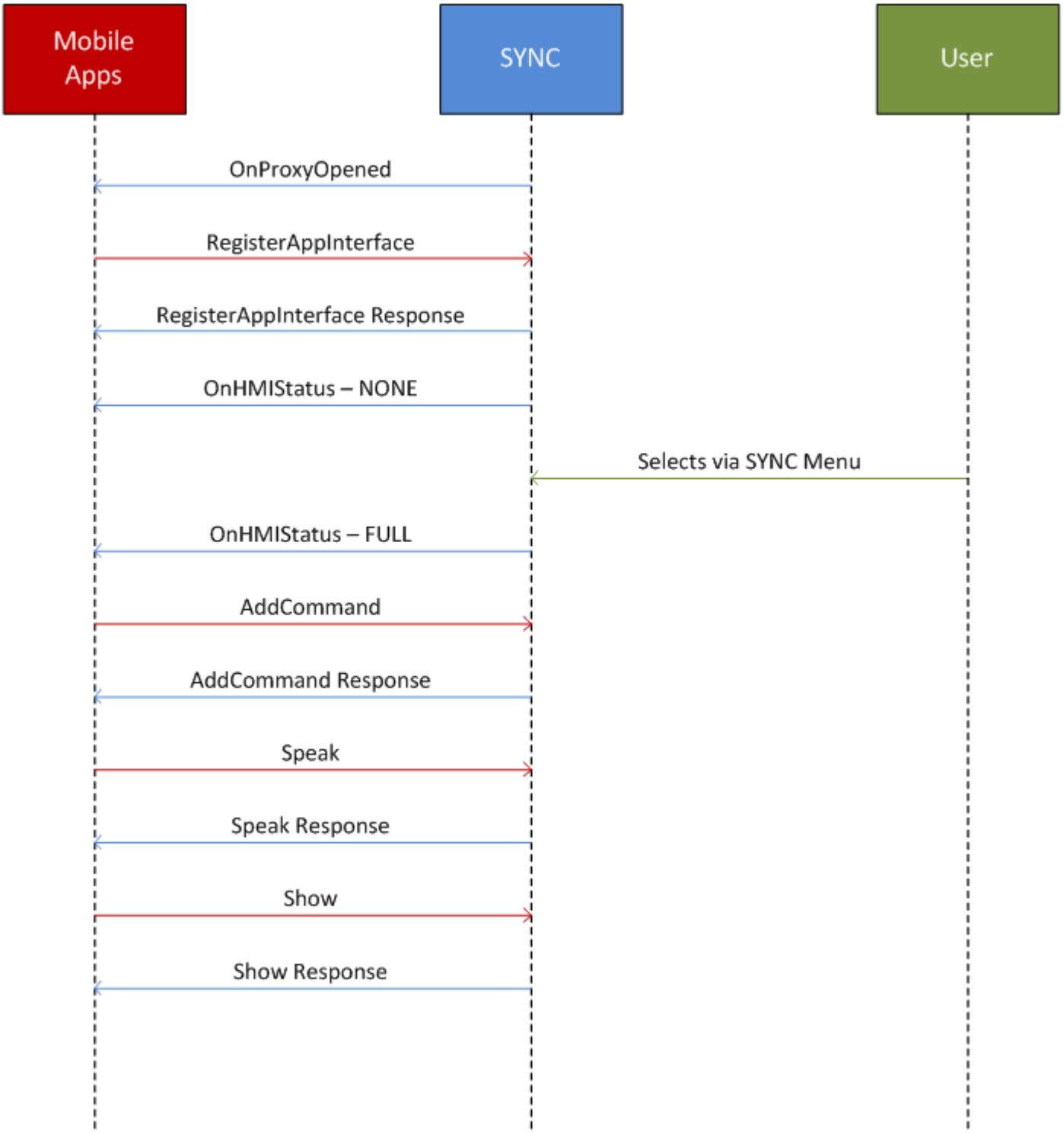

Figure 3: Typical RPC Exchange

6.3 RPC Example Application

Included in the SDK is an application called AppLinkTester, which has an example of using each RPC.

6.4 RPC Index

AddCommand	AddSubMenu	DeleteSubMenu
DeleteCommand	Alert	CreateInteractionChoiceSet
DeleteInteractionChoiceSet	RegisterAppInterface	UnregisterAppInterface
Speak	ResetGlobalProperties	SetGlobalProperties
SetMediaClockTimer	Show	SubscribeButton
UnsubscribeButton	PerformInteraction	EncodedSyncPData

6.5 AddCommand

Adds a Command to the application's Command Menu.

A Command will be added to the end of the list of elements in the Command Menu under the following conditions:

- When a Command is added with no MenuParams value provided
- When a MenuParams value is provided with a MenuParam.position value greater than or equal to the number of menu items currently defined in the menu specified by the MenuParam.parentID value

The set of choices which the application builds using AddCommand can be a mixture of:

- Choices having only VR synonym definitions, but no MenuParams definitions
- Choices having only MenuParams definitions, but no VR synonym definitions
- Choices having both MenuParams and VR synonym definitions

When a user initiates an interaction, the user may choose from whatever choices are defined at that moment. It is up to the application to ensure that all appropriate choices are defined before the user initiates an interaction.

Note: *A consequence of this is that it is possible for the application to be in the middle of adding commands (or submenu items) when the user presses the PTT or Menu button and only the commands added up to that moment will be available for the user to choose from (either by VR or Menu). It is possible in this situation for the user to not have had all intended options available when they started the interaction. The application will receive an OnHMIStatus with a SystemContext value of VRSESSION or MENU indicating that such a user-initiated interaction has begun. The application can make use of this notification to know that, if it is still building a Command Menu using AddCommand/AddSubMenu, the results of the interaction should be discarded. This approach is necessary because there is no way to "cancel" an interaction in progress (neither a user initiated interaction, nor an app-initiated interaction).*

Note: *AppLink™ batches together AddCommand and AddSubMenu requests before making them available for selection by the user via PTT or the menu. The end of a batch of successive AddCommand/AddSubMenu requests is defined as 500 milliseconds passing with no further AddCommand/AddSubMenu requests arriving at SYNC®. When the batch of requests has ended, SYNC® then prepares these commands for use by the user. A command should be available for use 1 second after the end of the batch in which that command was added (timing based batch size).*

There are a few other noteworthy consequences of MenuParams for a given command:

- Commands that do not have vrCommands associated with them will not be accessible by voice commands (when the user hits push-to-talk).
- Commands that do not have menuParams associated with them will not be accessible through the HMI application menu

6.5.1 HMI Status Requirements

Element	Requirement
HMILevel	FULL, LIMITED, or BACKGROUND

AudioStreamingState	N/A
SystemContext	Should not be attempted when VRSESSION or MENU

6.5.2 Request

Name	Type	Description	Req.	Notes	AppLink Ver. Available
cmdID	Int32	Unique ID that identifies the command. Is returned in an OnCommand notification to identify the command selected by the user.	Y	Min Value: 0 Max Value: 2000000000	AppLink 1.0
menuParams	MenuParams	If provided, this will define the command and how it is added to the Command Menu. Please see MenuParams enumeration for more details.	N	Optional only if vrCommand is provided	AppLink 1.0
vrCommands	String[]	If provided, defines one or more VR phrases the recognition of any of which triggers the OnCommand notification with this cmdID.	N	Optional only if menuParams is provided. If provided, array must contain at least one non-empty (not null, not zero-length, not whitespace only) element	AppLink 1.0

6.5.3 Response

Indicates that the corresponding request has failed or succeeded, if the response returns with a SUCCESS result code, this means a command was added to the Command Menu successfully.

Non-default Result Codes:

- INVALID_ID
- DUPLICATE_NAME

6.5.4 Related Operations

DeleteCommand
AddSubMenu
DeleteSubMenu

6.5.5 Example Function Call

```
commandReq = RPCRequestFactory.buildAddCommand(100, "Skip", new
Vector<String>(Arrays.asList(new String[] {"Skip"})), autoIncCorrID++);
_syncProxy.sendRPCRequest(commandReq);
```

6.6 AddSubMenu

Adds a SubMenu to the Command Menu (See the Programming Guide for a discussion of Command Menu). A SubMenu can only be added to the Top Level Menu (i.e. a SubMenu cannot be added to a SubMenu), and may only contain commands as children.

For a discussion of conflict with user-initiated interaction, see *AddCommand*.

6.6.1 HMI Status Requirements

Element	Requirement
HMILevel	FULL, LIMITED, or BACKGROUND
AudioStreamingState	N/A
SystemContext	Should not be attempted when VRSESSION or MENU

6.6.2 Request

Name	Type	Description	Req.	Notes	AppLink Ver. Available
menuID	Int32	Unique ID that identifies this sub menu. This value is used in AddCommand to which SubMenu is the parent of the command being added.	Y	Min Value: 0 Max Value: 2000000000	AppLink 1.0
position	Int16	Position within the items of the top level Command Menu. 0 will insert at the front, 1 will insert after the first existing element, etc. Position of any submenu will always be located before the return and exit options.	N	Min Value: 0 Max Value: 1000 • If position is greater or equal than the number of items on top level, the sub menu will be appended by the end. • If this parameter is omitted, the entry will be added at the end of the list.	AppLink 1.0
menuName	String	Text which is displayed representing this submenu item	Y		AppLink 1.0

6.6.3 Response

Indicates that the corresponding request either failed or succeeded. If the response returns with a SUCCESS result code, this means the SubMenu was added to the Command Menu successfully

Non-default Result Codes:

- INVALID_ID
- DUPLICATE NAME

6.6.4 Related Operations

DeleteSubMenu
AddCommand
DeleteCommand

6.6.5 Example Function Call

```
RPCMessage req;
req = RPCRequestFactory.buildAddSubMenu(menuID, menuName,autoIncCorrID++);
      _syncProxy.sendRPCRequest(req);
```

6.7 DeleteSubMenu

Deletes a submenu from the Command Menu.

For a discussion of conflict with user-initiated interaction, see *AddCommand*.

When an app deletes a submenu that has child commands, those child commands are also deleted.

6.7.1 HMI Status Requirements

Element	Requirement
HMILevel	FULL, LIMITED, or BACKGROUND
AudioStreamingState	N/A
SystemContext	Should not be attempted when VRSESSION or MENU

6.7.2 Request

Name	Type	Description	Req.	Notes	AppLink Ver. Available
menuID	Int32	Unique ID that identifies the SubMenu to be delete	Y	Min Value: 0 Max Value: 2000000000	AppLink 1.0

6.7.3 Response

Non-default Result Codes:

- INVALID_ID
- IN_USE

6.7.4 Related Operations

AddCommand
AddSubMenu
DeleteCommand

6.7.5 Example Function Call

```
RPCMessage req;
req = RPCRequestFactory.buildDeleteSubMenu(menuID, autoIncCorrID++);
_syncProxy.sendRPCRequest(req);
```

6.8 DeleteCommand

Removes a command from the Command Menu (See the Programming Guide for an explanation of Command Menu).

For a discussion of conflict with user-initiated interaction, see AddCommand.

6.8.1 HMI Status Requirements

Element	Requirement
HMILevel	FULL, LIMITED, or BACKGROUND
AudioStreamingState	N/A
SystemContext	Should not be attempted when VRSESSION or MENU

6.8.2 Request

Name	Type	Description	Req.	Notes	AppLink Ver. Available
cmdID	Int32	Unique ID that identifies the Command to be deleted from Command Menu	Y	Min Value: 0 Max Value: 2000000000	AppLink 1.0

6.8.3 Response

Indicates that the corresponding request either failed or succeeded. If the response returns with a SUCCESS result code, this means a command was removed from the Command Menu successfully.

Non-default Result Codes:

- INVALID_ID

6.8.4 Related Operations

AddCommand
AddSubMenu
DeleteSubMenu

6.8.5 Example Function Call

```
DeleteCommand req;
req = RPCRequestFactory.buildDeleteCommand(commandID, autoIncCorrID++);
_syncProxy.sendRPCRequest(req);
```

6.9 Alert

Provides information to the user using either TTS, the Display or both and can include a system-generated alert tone.

- The displayed portion of the Alert, if any, will persist until the specified timeout has elapsed, or the Alert is preempted.
- An Alert will preempt (abort) any AppLink Operation that is in-progress, except an already-in-progress Alert.
- An Alert cannot be preempted by any AppLink Operation.
- An Alert can be preempted by a user action (button push).
- An Alert will fail if it is issued while another Alert is in progress.

- Although each Alert parameter is optional, in fact each Alert request must supply at least one of the following parameters:
  - alertText1
  - alertText2
  - ttsChunks

6.9.1 HMI Status Requirements

Element	Requirement
HMILevel	FULL
AudioStreamingState	Any
SystemContext	MAIN, MENU, VR **Note:** *When Alert is issued with MENU in effect, Alert is queued and "played" when MENU interaction is completed (i.e. SystemContext reverts to MAIN). When Alert is issued with VR in effect, Alert is queued and "played" when VR interaction is completed (i.e. SystemContext reverts to MAIN).* **Note:** *When both Alert and Speak are queued during MENU or VR, they are "played" back in the order in which they were queued, with all existing rules for "collisions" still in effect.*

6.9.2 Request

Name	Type	Description	Req.	Notes	AppLink Ver. Available
alertText1	String	Text to be displayed in the top field of the display during the Alert.	N	• Length is limited to what is indicated in *RegisterAppInterface* response. • If omitted, top display line will be cleared. • Text is always centered	AppLink 1.0
alertText1	String	Text to be displayed in the bottom field of the display during the Alert.	N	• Only permitted if HMI supports a second display line. • Length is limited to what is indicated in *RegisterAppInterface* response. • If omitted, bottom display line will be cleared. • Text is always centered	AppLink 1.0
ttsChunks	TTSChunk[]	Array of type TTSChunk which, taken together, specify what is to be spoken to the user.	N	• Array must have a least one element.	AppLink 1.0

duration	Int32	The duration of the displayed portion of the alert, in milliseconds. After this amount of time has passed, the display fields alertText1 and alertText2 will revert to what was displayed in those fields before the alert began	N	Min Value: 3000 Max Value: 10000 • If omitted, the default is 5000 milliseconds	AppLink 1.0
playTone	Boolean	Specifies whether the alert tone should be played before the TTS (if any) is spoken.	N	• If omitted, default is true.	AppLink 1.0

6.9.3 Response

If a resultCode of "SUCCESS" is returned, the request was accepted by SYNC. By the time the corresponding response is received, the Alert will have completed.

Non-default Result Codes:

- REJECTED
- ABORTED

6.9.4 Related Operations

Show
Speak

6.9.5 Example Function Call

```
Alert req;
req = RPCRequestFactory.buildAlert(ttsText, _mainInstance.logTag, alertText2,
playTone, duration, autoIncCorrID++);
_syncProxy.sendRPCRequest(req);
```

6.10 CreateInteractionChoiceSet

Creates a Choice Set which can be used in subsequent *PerformInteraction* Operations.

6.10.1 HMI Status Requirements

Element	Requirement
HMILevel	FULL, LIMITED, or BACKGROUND
AudioStreamingState	Any
SystemContext	MAIN, MENU, VR

6.10.2 Request

Name	Type	Description	Req.	Notes	AppLink Ver. Available
interactionChoiceSetID	Int32	A unique ID that identifies the Choice Set	Y	Min Value: 0 Max Value: 2000000000	AppLink 1.0

choiceSet	Choice[]	Array of one or more elements.	Y	Min Value: 1 Max Value: 100	AppLink 1.0

6.10.3 Response

Indicates that the corresponding request either failed or succeeded. If the response returns with a SUCCESS result code, this means the Choice Set was created.

Non-default Result Codes:

- INVALID_ID
- DUPLICATE NAME

6.10.4 Related Operations

DeleteInteractionChoiceSet
PerformInteraction

6.10.5 Example Function Call

```
RPCMessage req;
req = RPCRequestFactory.buildCreateInteractionChoiceSet(_choiceSet, choiceSetID,
autoIncCorrID++);
_syncProxy.sendRPCRequest(req);
```

6.11 DeleteInteractionChoiceSet

Deletes an existing Choice Set identified by the parameter interactionChoiceSetID. If the specified interactionChoiceSetID is currently in use by an active *PerformInteraction*, this call to delete the Choice Set will fail returning an IN_USE resultCode.

6.11.1 HMI Status Requirements

Element	Requirement
HMILevel	FULL, LIMITED, or BACKGROUND
AudioStreamingState	Any
SystemContext	MAIN, MENU, VR

6.11.2 Request

Name	Type	Description	Req.	Notes	AppLink Ver. Available
interactionChoiceSetID	Int32	A unique ID that identifies the Choice Set (specified in a previous call to *CreateInteractionChoiceSet*)	Y	Min Value: 0 Max Value: 2000000000	AppLink 1.0

6.11.3 Response

If a resultCode of "SUCCESS" is returned, the requested choice set has been created and can now be referenced by the application using the value of interactionChoiceSetID provided by the application.

Non-default Result Codes:
- INVALID_ID

6.11.4 Related Operations

CreateInteractionChoiceSet
PerformInteraction

6.11.5 Example Function Call

```
RPCMessage req;
req = RPCRequestFactory.buildDeleteInteractionChoiceSet(choiceSetID,
autoIncCorrID++);
      _syncProxy.sendRPCRequest(req);
```

6.12 RegisterAppInterface

Registers the application's interface with SYNC®, declaring properties of the registration, including the messaging interface version, the app name, etc. The mobile application must establish its interface registration with SYNC® before any other interaction with SYNC® can take place. The registration lasts until it is terminated either by the application calling the *UnregisterAppInterface* method, or by SYNC® sending an OnAppInterfaceUnregistered notification, or by loss of the underlying transport connection, or closing of the underlying message transmission protocol RPC session.

Until the application receives its first OnHMIStatus Notification, its HMI Status is assumed to be: HMILevel=NONE, AudioStreamingState=NOT_AUDIBLE, SystemContext=MAIN.

All SYNC® resources which the application creates or uses (e.g. Choice Sets, Command Menu, etc.) are associated with the application's interface registration. Therefore, when the interface registration ends, the SYNC® resources associated with the application are disposed of. As a result, even though the application itself may continue to run on its host platform (e.g. mobile device) after the interface registration terminates, the application will not be able to use the SYNC® HMI without first establishing a new interface registration and re-creating its required SYNC® resources. That is, SYNC® resources created by (or on behalf of) an application do not persist beyond the life-span of the interface registration.

Resources and settings whose lifespan is tied to the duration of an application's interface registration:
- Choice Sets
- Command Menus (built by successive calls to *AddCommand*)
- Media clock timer display value
- Media track display value
- Button subscriptions

The autoActivateID is used to grant an application the HMILevel and AudioStreamingState it had when it last disconnected.

Note: The autoActivateID parameter, and associated behavior, is currently ignored by SYNC®.

When first calling this method (i.e. first time within life cycle of mobile app), an autoActivateID should not be included. After successfully registering an interface, an autoActivateID is returned to the mobile application for it to use in subsequent connections. If the connection between SYNC® and the mobile application is lost, such as the vehicle is turned off while the application is running, the autoActivateID can then be passed in another call to RegisterAppInterface to re-acquire HMILevel=FULL.

If the application intends to stream audio it is important to indicate so via the isMediaApp parameter. When set to true, audio will reliably stream without any configuration required by the user. When not set, audio may stream, depending on what the user might have manually configured as a media source on SYNC®.

There is no time limit for how long the autoActivateID is "valid" (i.e. would confer focus and opt-in)

6.12.1 HMI Status Requirements

Element	Requirement
HMILevel	N/A
AudioStreamingState	N/A
SystemContext	N/A

6.12.2 Request

Name	Type	Description	Req.	Notes	AppLink Ver. Available
SYNC® MsgVersion	SYNC® MsgVersion	Declares what version of the SYNC® AppLink™ interface the application expects to use with SYNC®.	Y	To be compatible, app msg major version number must be less than or equal to SYNC® major version number. If msg versions are incompatible, app has 20 seconds to attempt successful RegisterAppInterface (w.r.t. msg version) on underlying protocol session, else will be terminated. Major version number is a compatibility declaration. Minor version number indicates minor functional variations (e.g. features, capabilities, bug fixes) when sent from SYNC® to app (in RegisterAppInterface response). However, the minor version number sent from the app to SYNC® (in RegisterAppInterface request) is ignored by SYNC®.	AppLink 1.0
appName	String	The mobile application's name. This name is displayed in the SYNC® Mobile Applications menu. It also serves as the unique identifier of the application for AppLink™ .	Y	- Must be 1-100 characters in length. - Must consist of following characters: - May not be the same (by case insensitive comparison) as the name or any synonym of any currently-registered application.	AppLink 1.0
ngnMediaScreenAppName	String	Provides an abbreviated version of the app name (if necessary) that will be displayed on the NGN media screen.	N	- Must be 1-5 characters - If not provided, value will be derived from appName truncated to 5 characters. - Must consist of following characters:	AppLink 1.0
vrSynonyms	String[]	An array of 1-100 elements, each element containing a voice-recognition synonym by which this app can be called when being addressed in the mobile applications menu.	N	- Each vr synonym is limited to 40 characters, and there can be 1-100 synonyms in array - May not be the same (by case insensitive comparison) as the name or any synonym of any currently-registered application. - Must consist of following characters:	AppLink 1.0
isMediaApplication	Boolean	Indicates that the application will be streaming audio to SYNC® (via A2DP) that is audible outside of the BT media source.	Y		AppLink 1.0
languageDesired	Language	An enumeration indicating what language the application intends to use for user interaction (Display, TTS and VR).	Y	- If the language indicated does not match the active language on SYNC®, the interface registration will be rejected. - If the user changes the SYNC® language while this interface registration is active, the interface registration will be terminated.	AppLink 1.0

autoActivateID	String	A value generated by SYNC® and returned in a previous RegisterAppInterface request. Used to restore previous HMILevel, and previous opt-in approval state of application.	N		AppLink 1.0

6.12.3 Response

The response message contains essential information about the SYNC system that the request was sent to. This information can be used by an application to alter functionality depending on which type of SYNC system it is running on. For instance, an application can use the displayCapabilities parameter to see whether the SYNC HMI offers a touch screen, a two-line display, or a one-line display.

The application could then use that information to make sure that the text written to the screen is readable for the given platform. If a resultCode other than SUCCESS is received, the application will have to send one or more RegisterAppInterface Requests until one has a result code of SUCCESS; otherwise, the application is not permitted to send any other Requests.

Non-default Result Codes:

- TOO_MANY_APPLICATIONS
- DUPLICATE_NAME
- APPLICATION_REGISTERED_ALREADY
- UNSUPPORTED_VERSION
- WRONG_LANGUAGE

Name	Type	Description	Req	Notes	AppLink Ver. Available
SYNC®MsgVersion	SYNC®MsgVersion	The SYNC® AppLink™ interface version which SYNC® is using.		To be compatible, app msg major version number must be less than or equal to SYNC® major version number. If msg versions are incompatible, app has 20 seconds to attempt successful RegisterAppInterface (w.r.t. msg version) on underlying protocol session, else will be terminated. Major version number is a compatibility declaration. Minor version number indicates minor functional variations (e.g. features, capabilities, bug fixes) when sent from SYNC® to app (in RegisterAppInterface response). However, the minor version number sent from the app to SYNC® (in RegisterAppInterface request) is ignored by SYNC®.	AppLink 1.0
autoActivateID	String	A value generated by SYNC® and used by the app in subsequent RegisterAppInterface requests to restore previous HMILevel, and previous opt-in approval state of application.			AppLink 1.0

language	Language	The language currently active on the SYNC® module.		If this language is not compatible with language requested by app, the operation will fail, but the returned language can be used on a subsequent RegisterAppInterface request.	AppLink 1.0
displayCapabilities	DisplayCapabilities	Describes the display capabilities of the connected SYNC® module.		These display capabilities are used by the application to determine how information can be displayed to the user.	AppLink 1.0
buttonCapabilities	ButtonCapabilities[]	Describes the HMI button capabilities of the connected SYNC® system.			AppLink 1.0
hmiZoneCapabilities	HmiZoneCapabilities[]	An array of HmiZoneCapabilities enumeration elements, indicating which HMI Zone Capabilities the connected SYNC® unit has.			AppLink 1.0
speechCapabilities	SpeechCapabilities[]	An array of SpeechCapabilities enumeration elements, indicating which speech capabilities the connected SYNC® unit has.			AppLink 1.0
vrCapabilities	VrCapabilities[]	An array of VrCapabilities enumeration elements, indicating which voice recognition (VR) capabilities the connected SYNC® unit has.			AppLink 1.0

6.12.4 Related Operations

UnregisterAppInterface
OnAppInterfaceUnregistered

6.12.5 Example Function Call

```
RegisterAppInterface req;
req = RPCRequestFactory.buildRegisterAppInterface(_mainInstance.logTag, false,
null);
_syncProxy.sendRPCRequest(req);
```

6.13 UnregisterAppInterface

Terminates an application's interface registration. This causes SYNC® to dispose of all resources associated with the application's interface registration (e.g. Command Menu items, Choice Sets, button subscriptions, etc.).

After the UnregisterAppInterface operation is performed, no other operations can be performed until a new app interface registration is established by calling *RegisterAppInterface*.

6.13.1 HMI Status Requirements

Element	Requirement
HMILevel	FULL, LIMITED, BACKGROUND, or

	NONE
AudioStreamingState	Any
SystemContext	Any

6.13.2 Request

This operation will fail if no interface registration is currently in effect (i.e. no *RegisterAppInterface* operation was performed prior to this call).

6.13.3 Response

Once this response is received, no other Operations will be accepted by SYNC® and no Notifications will be sent by SYNC®.

6.13.4 Related Operations

RegisterAppInterface
OnAppInterfaceUnregistered

6.13.5 Example Function Call

```
RPCMessage req;
req = RPCRequestFactory.buildUnregisterAppInterface(autoIncCorrID++);
_syncProxy.sendRPCRequest(req);
```

6.14 Speak

Speaks a phrase over the vehicle audio system using SYNC's TTS (text-to-speech) engine. The provided text to be spoken can be simply a text phrase, or it can consist of phoneme specifications to direct SYNC's TTS engine to speak a "speech-sculpted" phrase.

Receipt of the Response indicates the completion of the Speak operation, regardless of how the Speak operation may have completed (i.e. successfully, interrupted, terminated, etc.).

Requesting a new Speak operation while the application has another Speak operation already in progress (i.e. no corresponding Response for that in-progress Speak operation has been received yet) will terminate the in-progress Speak operation (causing its corresponding Response to be sent by SYNC) and begin the requested Speak operation.

Requesting a new Speak operation while the application has an *Alert* operation already in progress (i.e. no corresponding Response for that in-progress *Alert* operation has been received yet) will result in the Speak operation request being rejected (indicated in the Response to the Request).

Requesting a new *Alert* operation while the application has a Speak operation already in progress (i.e. no corresponding Response for that in-progress Speak operation has been received yet) will terminate the in-progress Speak operation (causing its corresponding Response to be sent by SYNC) and begin the requested *Alert* operation.

Requesting a new Speak operation while the application has a *PerformInteraction* operation already in progress (i.e. no corresponding Response for that in-progress *PerformInteraction* operation has been received yet) will result in the Speak operation request being rejected (indicated in the Response to the Request).

Requesting a *PerformInteraction* operation while the application has a Speak operation already in progress (i.e. no corresponding Response for that in-progress Speak operation has been received yet) will terminate the in-progress Speak operation (causing its corresponding Response to be sent by SYNC) and begin the requested *PerformInteraction* operation.

6.14.1 HMI Status Requirements

Element	Requirement
HMILevel	FULL
AudioStreamingState	Any
SystemContext	MAIN, MENU, VR **Note:** *When Alert is issued with MENU in effect, Alert is queued and "played" when MENU interaction is completed (i.e. SystemContext reverts to MAIN). When Alert is issued with VR in effect, Alert is queued and "played" when VR interaction is completed (i.e. SystemContext reverts to MAIN).* **Note:** *When both Alert and Speak are queued during MENU or VR, they are "played" back in the order in which they were queued, with all existing rules for "collisions" still in effect.*

6.14.2 Request

Name	Type	Description	Req.	Notes	AppLink Ver. Available
ttsChunks	TTSChunk[]	An array of 1-100 TTSChunk structs which, taken together, specify the phrase to be spoken.	Y	• The array must have 1-100 elements. • The total length of the phrase composed from the ttsChunks provided must be less than 500 characters or the request will be rejected. • Each chunk can be no more than 500 characters.	AppLink 1.0

6.14.3 Response

This Response notifies the application of the completion, interruption, or failure of a Speak Request.

Non-default Result Codes:

- REJECTED
- ABORTED

6.14.4 Related Operations

Alert

6.14.5 Example Function Call

```
Speak req;
req = RPCRequestFactory.buildSpeak(ttsText, autoCorrIncID++);
_syncProxy.sendRPCRequest(req);
```

6.15 ResetGlobalProperties

Resets the passed global properties to their default values as defined by SYNC.

The HELPPROMPT global property default value is generated by SYNC consists of the first vrCommand of each Command Menu item defined at the moment PTT is pressed.

The TIMEOUTPROMPT global property default value is the same as the HELPPROMPT global property default value.

6.15.1 HMI Status Requirements

Element	Requirement
HMILevel	FULL, LIMITED, or BACKGROUND
AudioStreamingState	Any
SystemContext	Any

6.15.2 Request

Name	Type	Description	Req	Notes	AppLink Ver. Available
properties	GlobalProperty[]	An array of one or more GlobalProperty enumeration elements indicating which global properties to reset to their default value.	Y	Array must have at least one element	AppLink 1.0

6.15.3 Response

Indicates whether the Global Properties were successfully set to their default values.

6.15.4 Related Operations

SetGlobalProperties

6.15.5 Example Function Call

```
ResetGlobalProperties req = new ResetGlobalProperties();
req.setCorrelationID(autoIncCorrID++);
Vector<GlobalProperty> properties = new Vector<GlobalProperty>();
properties.add(HELPPROMPT):
req.setProperties(properties);

_syncProxy.sendRPCRequest(req);
```

6.16 SetGlobalProperties

Sets value(s) for the specified global property(ies).

6.16.1 HMI Status Requirements

Element	Requirement
HMILevel	FULL, LIMITED, or BACKGROUND
AudioStreamingState	Any
SystemContext	Any

6.16.2 Request

Name	Type	Description	Req.	Notes	AppLink Ver. Available

helpPrompt	TTSChunk[]	Array of one or more TTSChunk elements specifying the help prompt used in an interaction started by PTT.	N	• Array must have at least one element • Only optional it timeoutPrompt has been specified	AppLink 1.0
timeoutPrompt	TTSChunk[]	Array of one or more TTSChunk elements specifying the help prompt used in an interaction started by PTT.	N	• Array must have at least one element • Only optional it helpPrompt has been specified	AppLink 1.0

6.16.3 Response

Indicates whether the requested Global Properties were successfully set.

6.16.4 Related Operations

ResetGlobalProperties

6.16.5 Example Function Call

```
RPCMessage req;
req = RPCRequestFactory.buildSetGlobalProperties (helpText, timeoutText,
autoIncCorrID++);

_syncProxy.sendRPCRequest(req);
```

6.17 SetMediaClockTimer

Sets the media clock/timer value and the update method (e.g. count-up, count-down, etc.).

6.17.1 HMI Status Requirements

Element	Requirement
HMILevel	FULL, LIMITED, or BACKGROUND
AudioStreamingState	Any
SystemContext	MAIN (queued during VR and MENU)

6.17.2 Request

Name	Type	Description	Req.	Notes	AppLink Ver. Available
startTime	StartTime	StartTime struct specifying hour, minute, second values to which media clock timer is set.	N	• If "updateMode" is COUNTUP or COUNTDOWN, this parameter must be provided. • Will be ignored for "PAUSE"/"RESUME"	AppLink 1.0
updateMode	UpdateMode	Specifies how the media clock/timer is to be updated (COUNTUP/COUNTDOWN/PAUSE/RESUME), based at the startTime.	Y	• When updateMode is PAUSE or RESUME, the start time value is ignored. • When updateMode is RESUME, the timer resumes counting from the timer's value when it was paused.	AppLink 1.0

6.17.3 Response

Non-default Result Codes:

- REJECTED
- IGNORED

6.17.4 Example Function Calls

```
RPCMessage req;
req = RPCRequestFactory.buildSetMediaClockTimer (0,15,30, updateMode,
autoIncCorrID++);
_syncProxy.sendRPCRequest(req);
```

6.18 Show

Updates the application's display text area, regardless of whether or not this text area is visible to the user at the time of the request. The application's display text area remains unchanged until updated by subsequent calls to Show.

The content of the application's display text area is visible to the user when the application's HMILevel is FULL or LIMITED, and the SystemContext=MAIN and no *Alert* is in progress.

The Show operation cannot be used to create an animated scrolling screen. To avoid distracting the driver, Show commands cannot be issued more than once every 4 seconds. Requests made more frequently than this will be rejected.

For more information on how Show information is presented on different displays, see Ze document.

6.18.1 HMI Status Requirements

Element	Requirement
HMILevel	FULL, LIMITED, or BACKGROUND
AudioStreamingState	Any
SystemContext	Any

6.18.2 Request

Name	Type	Description	Req.	Notes	AppLink Ver. Available
mainField1	String	Text to be displayed in a single-line display, or in the upper display line in a two-line display.	N	• If this parameter is omitted, the text of mainField1 does not change. • If this parameter is an empty string, the field will be cleared.	AppLink 1.0

mainField2	String	Text to be displayed on the second display line of a two-line display.	N	• If this parameter is omitted, the text of mainField2 does not change. • If this parameter is an empty string, the field will be cleared. • If provided and the display is a single-line display, the parameter is ignored.	AppLink 1.0
alignment	TextAlignment	Specifies how mainField1 and mainField2 text should be aligned on display.	N	• Applies only to mainField1 and mainField2 provided on this call, not to what is already showing in display • If this parameter is omitted, text in both mainField1 and mainField2 will be centered • Has no effect with navigation display	AppLink 1.0
statusBar	String	The text is placed in the status bar area.	N	**Note:** *The status bar only exists on navigation displays* • If this parameter is omitted, the status bar text will remain unchanged. • If this parameter is an empty string, the field will be cleared. • If provided and the display has no status bar, this parameter is ignored.	AppLink 1.0
mediaClock	String	Sets the value for the MediaClock field using a format described in the MediaClockFormat enumeration.	N	• Must be properly formatted as described in the MediaClockFormat enumeration • If a value of five spaces is provided, this will clear that field on the display (i.e. the media clock timer field will not display anything)	AppLink 1.0
mediaTrack	String	Text to be displayed in the track field.	N	• If parameter is omitted, the track field remains unchanged • If an empty string is provided, the field will be cleared • This field is only valid for media applications on navigation displays.	AppLink 1.0

6.18.3 Response

Non-default Result Codes:

- REJECTED

6.18.4 Related Operations

Alert
SetMediaClockTimer

6.18.5 Example Function Call

```
Show req;
req = RPCRequestFactory.buildShow(mainText1, mainText2, null, mediaClock,
mediaTrack, alignment, autoIncCorrID++);
_syncProxy.sendRPCRequest(req);
```

6.19 SubscribeButton

Establishes a subscription to button notifications for HMI buttons. Buttons are not necessarily physical buttons, but can also be "soft" buttons on a touch screen, depending on the display in the vehicle. Once subscribed to a particular button, an application will receive both OnButtonEvent and OnButtonPress notifications whenever that button is pressed. The application may also unsubscribe from notifications for a button by invoking the UnsubscribeButton operation.

When a button is depressed, an OnButtonEvent notification is sent to the application with a ButtonEventMode of BUTTONDOWN. When that same button is released, an OnButtonEvent notification is sent to the application with a ButtonEventMode of BUTTONUP.

When the duration of a button depression (that is, time between depression and release) is less than two seconds, an OnButtonPress notification is sent to the application (at the moment the button is released) with a ButtonPressMode of SHORT. When the duration is two or more seconds, an OnButtonPress notification is sent to the application (at the moment the two seconds have elapsed) with a ButtonPressMode of LONG.

The purpose of OnButtonPress notifications is to allow for programmatic detection of long button presses similar to those used to store presets while listening to the radio, for example.

When a button is depressed and released, the sequence in which notifications will be sent to the application is as follows:

For short presses:

- OnButtonEvent (ButtonEventMode = BUTTONDOWN)
- OnButtonEvent (ButtonEventMode = BUTTONUP)
- OnButtonPress (ButtonPressMode = SHORT)

For long presses:

- OnButtonEvent (ButtonEventMode = BUTTONDOWN)
- OnButtonPress (ButtonPressMode = LONG)
- OnButtonEvent (ButtonEventMode = BUTTONUP)

6.19.1 HMI Status Requirements

Element	Requirement
HMILevel	FULL, LIMITED, or BACKGROUND
AudioStreamingState	Any
SystemContext	Any

6.19.2 Request

Name	Type	Description	Req.	Notes	AppLink Ver. Available
buttonName	ButtonName	Name of the button to subscribe to	Y		AppLink 1.0

6.19.3 Response

Non-default Result Codes:
- UNSUPPORTED_BUTTON
- IGNORED

6.19.4 Related Operations

UnsubscribeButton

6.19.5 Example Function Call

```
SubscribeButton buttonReq;
buttonReq = RPCRequestFactory.buildSubscribeButton(ButtonName.OK, autoIncCorrID++);
_syncProxy.sendRPCRequest(buttonReq);
```

6.20 UnSubscribeButton

Deletes a subscription to button notifications for the specified button. For more information about button subscriptions, see *SubscribeButton*.

Application can unsubscribe from a button that is currently being pressed (i.e. has not yet been released), but app will not get button event.

6.20.1 HMI Status Requirements

Element	Requirement
HMILevel	FULL, LIMITED, or BACKGROUND
AudioStreamingState	Any
SystemContext	Any

6.20.2 Request

Name	Type	Description	Req.	Notes	AppLink Ver. Available
buttonName	ButtonName	Name of the button to unsubscribe from	Y		AppLink 1.0

6.20.3 Response

Non-default Result Codes:
- UNSUPPORTED_BUTTON
- IGNORED

6.20.4 Related Operations

SubscribeButton

6.20.5 Example Function Call

```
RPCMessage req;
req = RPCRequestFactory.buildUnsubscribeButton(ButtonName.OK, autoIncCorrID++);
_syncProxy.sendRPCRequest(req);
```

6.21 PerformInteraction

Performs an application-initiated interaction in which the user can select a Choice from among the specified Choice Sets. For instance, an application may use a PerformInteraction to ask a user to say the name of a song to play. The user's response is only valid if it appears in the specified Choice Sets and is recognized by SYNC..

6.21.1 HMI Status Requirements

Element	Requirement
HMILevel	FULL
AudioStreamingState	N/A
SystemContext	Requires MAIN, Is queued in VR or MENU.

6.21.2 Request

Name	Type	Description	Req.	Notes	AppLink Ver. Available
initialText	String	Displayed when the interaction begins. This text may be overlaid by the "Listening" prompt during the interaction. Text is displayed on first line of multiline display, and is centered. If text does not fit on line, it will be truncated.	Y		AppLink 1.0
initialPrompt	TTSChunk[]	An array of one or more TTSChunks that, taken together, specify what is to be spoken to the user at the start of an interaction.	Y		AppLink 1.0
interactionMode	InteractionMode	Indicates how user selects interaction choice. User can choose either by voice (VR_ONLY), by visual selection from the menu (MANUAL_ONLY), or by either mode (BOTH).	Y		AppLink 1.0
interactionChoiceSetIDList	Int32[]	Array of one or more Choice Set IDs. User can select any choice from any of the specified Choice Sets.	Y	Min Value: 0 Max Value: 2000000000	AppLink 1.0
helpPrompt	TTSChunk[]	An array of TTSChunks which, taken together, specify the help phrase to be spoken when the user says "help" during the VR session. If this parameter is omitted, the help prompt will be constructed by SYNC from the first vrCommand of each choice of all the Choice Sets specified in the interactionChoiceSetIDList parameter. **Note:** *The helpPrompt specified in SetGlobalProperties is not used by PerformInteraction.*	N		AppLink 1.0
timeoutPrompt	TTSChunk[]	An array of TTSChunks which, taken together, specify the phrase to be spoken when the listen times out during the VR session. If this parameter is omitted, the timeout prompt will be	N		AppLink 1.0

		the same as the help prompt (see helpPrompt parameter). **Note:** *The timeoutPrompt specified in SetGlobalProperties is not used by PerformInteraction.*			
timeout	Int32	The amount of time, in milliseconds, SYNC will wait for the user to make a choice (VR or Menu). If this time elapses without the user making a choice, the timeoutPrompt will be spoken. After this timeout value has been reached, the interaction will stop and a subsequent interaction will take place after SYNC speaks the timeout prompt. If that times out as well, the interaction will end completely. If omitted, the default is 10000ms.	N	Min Value: 5000 Max Value: 100000	AppLink 1.0

6.21.3 Response

Indicates that interaction is complete, or was rejected.

An interaction can complete when the user selects a choice, or when the interaction times out (no user selection was made), or when the interaction was interrupted by the user (pressing PTT or Menu button), or by system events (e.g. phone call), etc.

This is sent to the mobile application after the user makes a choice from the PerformInteraction Request; or if the user fails to make a selection and the Request times out. Two additional parameters, cmdID and triggerSource, are provided with this Response. The cmdID parameter represents the ID of the Choice in the InteractionChoices that was selected by the user. The triggerSource parameter indicates where said Choice selection came from (either Menu or VR).

Non-default Result Codes:

- INVALID_ID
- DUPLICATE_NAME
- REJECTED
- ABORTED

Name	Type	Description	Req.	Notes	AppLink Ver. Available
choiceID	Int32	Choice ID of the item that was selected (verbally or via menu) by the user from the interactionChoiceSetIDList	N		AppLink 1.0
triggerSource	TriggerSource	Indicates whether choice was selected via VR or via a menu selection (using the OK button).	Y		AppLink 1.0

6.21.4 Related Operations

CreateInteractionChoiceSet
DeleteInteractionChoiceSet

6.21.5 Example Function Call

```
RPCMessage req;
req = RPCRequestFactory.buildPerformInteraction(initPrompt, displayText,
interactionChoiceSetIDList, helpPrompt, timeoutPrompt, interactionMode, timeout,
autoIncCorrID++);
_syncProxy.sendRPCRequest(req);
```

6.22 EncodedSyncPData

Sends base64 encoded SyncP packets to the SYNC module.

6.22.1 HMI Status Requirements

Element	Requirement
HMILevel	FULL, LIMITED, or BACKGROUND
AudioStreamingState	Any
SystemContext	Any

6.22.2 Request

Name	Type	Description	Req.	Notes	AppLink Ver. Available
data	String[]	An array of 1-100 elements, each element containing a base64 encoded SyncP packet.	Y	The maximum length of each array element is 10000 characters.	AppLink 1.0

6.22.3 Response

Non-default Result Codes:

6.22.4 Related Operations

OnEncodedSyncPData

6.22.5 Example Function Call

```
EncodedSyncPData msg = new EncodedSyncPData();
msg.setData(data);
msg.setCorrelationID(autoIncCorrID++);

_syncProxy.sendRPCRequest(msg);
```

7 AppLink™ Notifications

Notifications indicate to the application that an event has happened on SYNC. These events include things such as: the user pressing a button, user-initiated interactions completing, a change in HMILevel or AudioStreamingState, etc. The notification indicates that the event has occurred, and provides data associated with that event (e.g. an interaction selection, button name, etc.).

The HMI Status Requirements section for each notification indicates under which HMI Status conditions the notification can be received.

Notifications do not contain a correlationID because a notification does not correlate to any specific request.

7.1 Notification Index

OnButtonEvent	OnButtonPress	OnCommand
OnAppInterfaceUnregistered	OnHMIStatus	OnEncodedSYNCPData
OnTBTClientState	OnDriverDistraction	GenericResponse

7.2 OnButtonEvent

Notifies application that user has depressed or released a button to which the application has subscribed.

Further information about button events and button-presses can be found at *SubscribeButton*.

7.2.1 HMI Status Requirements

Element	Requirement
HMILevel	• The application will receive OnButtonEvent notifications for all subscribed buttons when HMILevel is FULL. • The application will receive OnButtonEvent notifications for subscribed media buttons when HMILevel is LIMITED. • Media buttons include SEEKLEFT, SEEKRIGHT, TUNEUP, TUNEDOWN, and PRESET_0-PRESET_9. • The application will not receive OnButtonEvent notification when HMILevel is BACKGROUND.
AudioStreamingState	Any
SystemContext	MAIN, VR. In MENU, only PRESET buttons. In VR, pressing any subscribable button will cancel VR.

7.2.2 Parameter List

Name	Type	Description	Req.	Notes	AppLink Ver. Available
buttonName	ButtonName	Name of the button which triggered this event.			AppLink 1.0
buttonEvent Mode	ButtonEventMode	Indicates button was depressed (DOWN) or released (UP).			AppLink 1.0

7.2.3 Related Operations

SubscribeButton
UnsubscribeButton

7.2.1 Example Function Call

```
@Override
public void onOnButtonEvent(OnButtonEvent notification) {
	if (notification.getButtonEventMode() == BUTTONDOWN ) {
		if (notification.getButtonName() == SEEKRIGHT) {
			_audioDetails.skipAudioPlayer();
		}
	}
}
```

7.3 OnButtonPress

Notifies application of button press events for buttons to which the application is subscribed. SYNC supports two button press events defined as follows:

- SHORT - Occurs when a button is depressed, then released within two seconds. The event is considered to occur immediately after the button is released.
- LONG - Occurs when a button is depressed and held for two seconds or more. The event is considered to occur immediately after the two second threshold has been crossed, before the button is released

7.3.1 HMI Status Requirements

Element	Requirement
HMILevel	• The application will receive OnButtonPress notifications for all subscribed buttons when HMILevel is FULL. • The application will receive OnButtonPress notifications for subscribed media buttons when HMILevel is LIMITED. • Media buttons include SEEKLEFT, SEEKRIGHT, TUNEUP, TUNEDOWN, and PRESET_0-PRESET_9. • The application will not receive OnButtonPress notification when HMILevel is BACKGROUND or NONE.
AudioStreamingState	Any
SystemContext	MAIN, VR. In MENU, only PRESET buttons. In VR, pressing any subscribable button will cancel VR.

7.3.2 Parameter List

Name	Type	Description	Req.	Notes	AppLink Ver. Available
buttonName	ButtonName	Name of the button which triggered this event.			AppLink 1.0
buttonPress Mode	ButtonPressMode	Indicates whether this is an SHORT or LONG button press event.			AppLink 1.0

7.3.3 Related Operations

SubscribeButton
UnsubscribeButton

7.3.4 Example Function Call

```
@Override
public void onOnButtonPress(OnButtonPress notification) {
	if(notification.getButtonName() == OK)
```

```
        {
                _audioDetails.startAudioPlayer();
        }
}
```

7.4 OnCommand

This is called when a command was selected via VR after pressing the PTT button, or selected from the menu after pressing the MENU button.

Note: *Sequence of OnHMIStatus and OnCommand notifications for user-initiated interactions is indeterminate.*

7.4.1 HMI Status Requirements

Element	Requirement
HMILevel	FULL
AudioStreamingState	Any
SystemContext	Any

7.4.2 Parameter List

Name	Type	Description	Req.	Notes	AppLink Ver. Available
cmdID	Int32	The cmdID of the command the user selected. This is the cmdID value provided by the application in the *AddCommand* operation that created the command.		Min: 0 Max: 2000000000	AppLink 1.0
triggerSource	TriggerSource	Indicates whether command was selected via VR or via a menu selection (using the OK button).			AppLink 1.0

7.4.3 Related Operations

AddCommand
DeleteCommand
DeleteSubMenu

7.4.4 Example Function Call

```
@Override
public void onOnCommand(OnCommand notification) {
      if(notification.getCmdID() == 100)
      {
            _audioDetails.skipAudioPlayer();
      }
}
```

7.5 OnAppInterfaceUnregistered

Notifies an application that its interface registration has been terminated. This means that all SYNC resources associated with the application are discarded, including the Command Menu, Choice Sets, button subscriptions, etc.

For more information about SYNC resources related to an interface registration, see *RegisterAppInterface*.

7.5.1 HMI Status Requirements

Element	Requirement
HMILevel	NONE
AudioStreamingState	Any
SystemContext	Any

7.5.2 Parameter List

Name	Type	Description	Req.	Notes	AppLink Ver. Available
reason	AppInterfaceUnregisteredReason	The reason the application's interface registration was terminated.			AppLink 1.0

7.5.3 Related Operations

RegisterAppInterface

7.6 OnHMIStatus

Notifies an application that HMI conditions have changed for the application. This indicates whether the application can speak phrases, display text, perform interactions, receive button presses and events, stream audio, etc. This notification will be sent to the application when there has been a change in any one or several of the indicated states (HMILevel, AudioStreamingState or SystemContext) for the application.

All three values are, in principle, independent of each other (though there may be some relationships). A value for one parameter should not be interpreted from the value of another parameter.

There are no guarantees about the timeliness or latency of the OnHMIStatus notification. Therefore, for example, information such as AudioStreamingState may not indicate that the audio stream became inaudible to the user exactly when the OnHMIStatus notification was received.

7.6.1 Parameter List

Name	Type	Description	Req.	Notes	AppLink Ver. Available
hmiLevel	HMILevel	The current HMI Level in effect for the application.			AppLink 1.0
audioStreamingState	AudioStreamingState	Current state of audio streaming for the application. When this parameter has a value of NOT_AUDIBLE, the application must stop streaming audio to SYNC. Informs app whether any currently streaming audio is audible to user (AUDIBLE) or not (NOT_AUDIBLE). A value of NOT_AUDIBLE means that either the application's audio will not be audible to the user, or that the application's audio should not be audible to the user (i.e. some other application on the mobile device may be streaming audio and the application's audio would be blended with that other audio).			AppLink 1.0
systemContext	SystemContext	Indicates that a user-initiated interaction is in-progress (VRSESSION or MENU), or not (MAIN).			AppLink 1.0

7.6.2 Related Operations

RegisterAppInterface

7.7 OnEncodedSYNCPData

Sends an array of base64 encoded SyncP packets to the application.

7.7.1 HMI Status Requirements

Element	Requirement
HMILevel	Can be sent with FULL, LIMITED or BACKGROUND
AudioStreamingState	Any
SystemContext	Any

7.7.2 Parameter List

Name	Type	Description	Req.	Notes	AppLink Ver. Available
data	String[]	An array of 1-100 elements, each element containing a base64 encoded SyncP packet.		The maximum length of each array element is 10000 characters	AppLink 1.0

7.7.3 Related Operations

EncodedSYNCPData

7.7.4 Example Function Call

```
@Override
public void onOnEncodedSyncPData(OnEncodedSyncPData notification) {
      beginProcessingData(notification.getData());
}
```

7.8 OnTBTClientState

Notifies the application of the current TBT client status on the module.

7.8.1 HMI Status Requirements

Element	Requirement
HMILevel	Can be sent with FULL, LIMITED or BACKGROUND
AudioStreamingState	Any
SystemContext	Any

7.8.2 Parameter List

Name	Type	Description	Req.	Notes	AppLink Ver. Available
state	TBTState	Current state of TBT client.			AppLink 1.0

7.8.3 Related Operations

TBD

7.8.4 Example Function Call

```
@Override
public void onOnTBTClientState(OnTBTClientState notification) {
	if (notification.getState() == ROUTE_ACCEPTED) {
		beginNavigation();
	}
}
```

7.9 OnDriverDistraction

Notifies the application of the current driver distraction state (i.e. whether driver distraction rules are in effect, or not).

7.9.1 HMI Status Requirements

Element	Requirement
HMILevel	Can be sent with FULL, LIMITED or BACKGROUND
AudioStreamingState	Any
SystemContext	Any

7.9.2 Parameter List

Name	Type	Description	Req.	Notes	AppLink Ver. Available
state	DriverDistractionState	Current driver distraction state (i.e. whether driver distraction rules are in effect, or not).			AppLink 1.0

7.9.3 Related Operations

TBD

7.9.4 Example Function Call

```
@Override
public void onOnDriverDistraction(OnDriverDistraction notification) {
	if(notification.getState() == DD_OFF) {
		speak("User Commands Enabled");
	}
}
```

7.10 GenericResponse

Notifies an application that a request it issued was not a recognized RPC request, and no corresponding response could be sent.

7.10.1 Nondefault Result Code

INVALID_DATA

8 Structures

Definitions of structures used in operations and notifications.

8.1 Structures Index

ButtonCapabilities	Choice	Command
DisplayCapabilities	MenuParams	StartTime
SYNCMsgVersion	TextField	TTSChunk

8.2 ButtonCapabilities

Provides information about the capabilities of a SYNC HMI button.

8.2.1 Parameter List

Name	Type	Description	AppLink Ver. Available
name	ButtonName	The name of theSYNC HMI button.	AppLink 1.0
shortPressAvailable	Boolean	The button supports a SHORT press. See ButtonPressMode for more information.	AppLink 1.0
longPressAvailable	Boolean	The button supports a LONG press. See ButtonPressMode for more information.	AppLink 1.0
upDownAvailable	Boolean	The button supports "button down" and "button up". When the button is depressed, the OnButtonEvent notification will be invoked with a value of BUTTONDOWN. When the button is released, the OnButtonEvent notification will be invoked with a value of BUTTONUP.	AppLink 1.0

8.3 Choice

A choice is an option which a user can select either via the menu or via voice recognition (VR) during an application initiated interaction.

8.3.1 Parameter List

Name	Type	Description	AppLink Ver. Available
choiceID	Int16	Application-scoped identifier that uniquely identifies this choice. Min: 0 Max: 65535	AppLink 1.0
menuName	String	Text which appears in menu, representing this choice. Min: 1 Max: 100	AppLink 1.0
vrCommands	String[]	An array of strings to be used as VR synonyms for this choice. If this array is provided, it must have at least one non-empty element	AppLink 1.0

8.4 Command

A command is an option which a user can select either via the menu or via voice recognition (VR) during a user-initiated interaction.

8.4.1 Parameter List

Name	Type	Description	AppLink Ver. Available
choiceID	Int16	Application-scoped identifier that uniquely identifies this choice. Min: 0 Max: 65535	AppLink 1.0
menuName	String	Text which appears in menu, representing this choice.	
vrCommands	String[]	An array of strings to be used as VR synonyms for this choice. If this array is provided, it must have at least one non-empty element	

8.5 DisplayCapabilities

Contains information about the display for the SYNC system to which the application is currently connected.

8.5.1 Parameter List

Name	Type	Description	AppLink Ver. Available
displayType	DisplayType	The type of display	AppLink 1.0
textFields	TextField[]	An array of TextField structures, each of which describes a field in the HMI which the application can write to using operations such as Show, *SetMediaClockTimer*, etc. This array of TextField structures identify all the text fields to which the application can write on the current display (identified by DisplayType).	AppLink 1.0
mediaClockFormats	MediaClockFormat[]	An array of MediaClockFormat elements, defining the valid string formats used in specifying the contents of the media clock field	AppLink 1.0

8.6 MenuParams

Used when adding a sub menu to an application menu or existing sub menu.

8.6.1 Parameter List

Name	Type	Description	AppLink Ver. Available
parentID	Int32	The unique ID of an existing submenu to which a command will be added. If this element is not provided, the command will be added to the top level of the Command Menu. • Min: 0 • Max: 2000000000	AppLink 1.0
position	Int16	Position within the items of the parent Command Menu. 0 will insert at the front, 1 will insert after the first existing element, etc. Position of any submenu will always be located before the return and exit options. • Min Value: 0 • Max Value: 1000 • If position is greater or equal than the number of items in the parent Command Menu, the sub menu will be appended to the end of that Command Menu. • If this element is omitted, the entry will be added at the end of the parent menu.	AppLink 1.0
menuName	String	Text which appears in menu, representing this command. • Min: 1	AppLink 1.0

		• Max: 100	

8.7 StartTime

Describes the hour, minute and second values used to set the media clock.

8.7.1 Parameter List

Name	Type	Description	AppLink Ver. Available
hours	Int16	The hour. Minvalue="0", maxvalue="59" **Note:** *Some display types only support a max value of 19. If out of range, it will be rejected.*	AppLink 1.0
minutes	Int16	The minute. Minvalue="0", maxvalue="59".	AppLink 1.0
seconds	Int16	The second. Minvalue="0", maxvalue="59".	AppLink 1.0

8.8 SYNCMsgVersion

Specifies the version number of the SYNC V4 interface. This is used by both the application and SYNC to declare what interface version each is using.

8.8.1 Parameter List

Name	Type	Description	AppLink Ver. Available
majorVersion	Int16	• minvalue="1" • maxvalue="10"	AppLink 1.0
minorVersion	Int16	• minvalue="0" • maxvalue="1000"	AppLink 1.0

8.9 TextField

Struct defining the characteristics of a displayed field on the HMI.

8.9.1 Parameter List

Name	Type	Description	AppLink Ver. Available
name	TextFieldName	Enumeration identifying the field.	AppLink 1.0
characterSet	CharacterSet	The character set that is supported in this field.	AppLink 1.0
width	Int16	The number of characters in one row of this field. • Minvalue="1" • maxvalue="40"	AppLink 1.0
rows	Int16	The number of rows for this text field. • Minvalue="1" • maxvalue="3".	AppLink 1.0

8.10 TTSChunk

Specifies what is to be spoken. This can be simply a text phrase, which SYNC will speak according to its own rules. It can also be phonemes from either the Microsoft SAPI phoneme set, or from the LHPLUS phoneme set. It can also be a pre-recorded sound in WAV format (either developer-defined, or provided by the SYNC platform).

In SYNC, words, and therefore sentences, can be built up from phonemes and are used to explicitly provide the proper pronounciation to the TTS engine. For example, to have SYNC pronounce the word "read" as "red", rather than as when it is pronounced like "reed", the developer would use phonemes to express this desired pronounciation.

For more information about phonemes, see http://en.wikipedia.org/wiki/Phoneme.

8.10.1 Parameter List

Name	Type	Description	AppLink Ver. Available
text	String	Text to be spoken, or a phoneme specification, or the name of a pre-recorded sound. The contents of this field are indicated by the "type" field.	AppLink 1.0
type	SpeechCapabilities	Indicates the type of information in the "text" field (e.g. phrase to be spoken, phoneme specification, name of pre-recorded sound).	AppLink 1.0

9 Enumerations

9.1 Enumeration Index

Result	ButtonPressMode	ButtonEventMode
Language	UpdateMode	InteractionMode
TriggerSource	HMILevel	AudioStreamingState
SystemContext	AppInterfaceUnregisteredReason	HMIZoneCapabilities
SpeechCapabilities	VRCapabilities	ButtonName
MediaClockFormat	DisplayType	CharacterSet
TextFieldName	TextAlignment	GlobalProperty
TBTState	DriverDistractionState	

9.2 Result

Defines the possible result codes returned by SYNC to the application in a Response to a requested operation

9.2.1 Parameter List

Name	Description	AppLink Ver. Available
SUCCESS	The request succeeded.	AppLink 1.0
INVALID_DATA	• The data sent is invalid. For example: • Invalid Json syntax • Parameters out of bounds (number or enum range) • Mandatory parameters not provided • Parameter provided with wrong type • Invalid characters • Empty string	AppLink 1.0
UNSUPPORTED_REQUEST	The request is not supported by SYNC.	AppLink 1.0
OUT_OF_MEMORY	The system could not process the request because the necessary memory couldn't be allocated.	AppLink 1.0
TOO_MANY_PENDING_REQUESTS	There are too many requests pending (means that the response has not been delivered yet). There is a limit of 1000 pending requests at a time	AppLink 1.0
INVALID_ID	One of the provided IDs is not valid. For example: • CorrelationID • CommandID • MenuID	AppLink 1.0
DUPLICATE_NAME	The provided name or synonym is a duplicate of some already-defined name or synonym.	AppLink 1.0
TOO_MANY_APPLICATIONS	There are already too many registered applications.	AppLink 1.0
APPLICATION_ALREADY_REGISTERED	Specified application name is already associated with an active interface registration. Attempts at doing a second *RegisterAppInterface* on a given protocol session will also cause this.	AppLink 1.0
UNSUPPORTED_VERSION	SYNC does not support the interface version requested by the mobile application.	AppLink 1.0
WRONG_LANGUAGE	The requested language is currently not supported. Might be because of a mismatch of the currently active language	AppLink 1.0

	on SYNC and the requested language.	
APPLICATION_NOT_REGISTERED	The request cannot be executed because no application interface has been registered via *RegisterAppInterface*.	AppLink 1.0
IN_USE	The data may not be changed, because it is currently in use. For example, when trying to delete a Choice Set that is currently involved in an interaction.	AppLink 1.0
SUBSCRIBED_ALREADY	There is already an existing subscription for this item.	AppLink 1.0
REJECTED	The requested operation was rejected. No attempt was made to perform the operation	AppLink 1.0
ABORTED	The requested operation was aborted due to some pre-empting event (e.g. button push, Alert pre-empts Speak, etc.).	AppLink 1.0
IGNORED	The requested operation was ignored because it was determined to be redundant (e.g. pause media clock when already paused).	AppLink 1.0
UNSUPPORTED_BUTTON	A button that was requested for subscription is not supported on the currently connected SYNC platform. See DisplayCapabilities for further information on supported buttons on the currently connected SYNC platform.	AppLink 1.0
FILE_NOT_FOUND	The specified file could not be found on SYNC.	AppLink 1.0
GENERIC_ERROR	Problem encountered in SYNC AppLink	AppLink 1.0

9.2.2 Example Use

```
Result.SUCCESS
```

9.3 ButtonPressMode

Indicates whether this is a LONG or SHORT button press.

9.3.1 Parameter List

Name	Description	AppLink Ver. Available
LONG	The button has been depressed for 2 seconds. The button may remain depressed after receiving this event	AppLink 1.0
SHORT	The button was released before the 2-second long-press interval had elapsed.	AppLink 1.0

9.3.2 Example Use

```
ButtonPressMode.LONG
```

9.4 ButtonEventMode

Indicates whether the button was depressed or released. A BUTTONUP event will always be preceded by a BUTTONDOWN event.

9.4.1 Parameter List

Name	Description	AppLink Ver. Available
BUTTONUP	The button was released.	AppLink 1.0
BUTTONDOWN	The button was depressed.	AppLink 1.0

9.4.2 Example Use

```
ButtonEventMode.BUTTONUP
```

9.5 Language

Specifies the language to be used for TTS, VR, displayed messages/menus

9.5.1 Parameter List

Name	Description	AppLink Ver. Available
EN-US	US English	AppLink 1.0
ES-MX	Mexican Spanish	AppLink 1.0
FR-CA	French Canadian	AppLink 1.0

9.5.2 Example Use

```
Language.EN_US
```

9.6 UpdateMode

Specifies what function should be performed on the media clock/counter.

9.6.1 Parameter List

Name	Description	AppLink Ver. Available
COUNTUP	Starts the media clock timer counting upward, in increments of 1 second.	AppLink 1.0
COUNTDOWN	Starts the media clock timer counting downward, in increments of 1 second.	AppLink 1.0
PAUSE	Pauses the media clock timer.	AppLink 1.0
RESUME	Resumes the media clock timer. The timer resumes counting in whatever mode was in effect before pausing (i.e. COUNTUP or COUNTDOWN).	AppLink 1.0

9.6.2 Example Use

```
UpdateMode.COUNTUP
```

9.7 InteractionMode

For application-initiated interactions (*PerformInteraction*), this specifies the mode by which the user is prompted and by which the user's selection is indicated.

9.7.1 Parameter List

Name	Description	AppLink Ver. Available
MANUAL_ONLY	This mode causes the interaction to occur only on the display, meaning the choices are presented and selected only via the display. Selections are viewed with the SEEKRIGHT, SEEKLEFT, TUNEUP, TUNEDOWN buttons. User's selection is indicated with the OK button.	AppLink 1.0
VR_ONLY	This mode causes the interaction to occur only through TTS and VR. The user is prompted via TTS to select a choice by saying one of the choice's synonyms.	AppLink 1.0
BOTH	This mode is a combination of MANUAL_ONLY and VR_ONLY, meaning the user is prompted both visually and audibly. The user can make a selection either	AppLink 1.0

	using the mode described in MANUAL_ONLY or using the mode described in VR_ONLY. If the user views selections as described in MANUAL_ONLY mode, the interaction becomes strictly, and irreversibly, a MANUAL_ONLY interaction (i.e. the VR session is cancelled, although the interaction itself is still in progress). If the user interacts with the VR session in any way (e.g. speaks a phrase, even if it is not a recognized choice), the interaction becomes strictly, and irreversibly, a VR_ONLY interaction (i.e. the MANUAL_ONLY mode forms of interaction will no longer be honored). The TriggerSource parameter of the *PerformInteraction* response will indicate which interaction mode the user finally chose to attempt the selection (even if the interaction did not end with a selection being made).	

9.7.2 Example Use

```
InteractionMode.VR_ONLY
```

9.8 TriggerSource

Indicates whether choice/command was selected via VR or via a menu selection (using SEEKRIGHT/SEEKLEFT, TUNEUP, TUNEDOWN and OK buttons).

9.8.1 Parameter List

Name	Description	AppLink Ver. Available
MENU	Selection made via menu (i.e. using SEEKRIGHT/SEEKLEFT, TUNEUP, TUNEDOWN and OK buttons).	AppLink 1.0
VR	Selection made via VR session.	AppLink 1.0

9.8.2 Example Use

```
TriggerSource.TS_MENU
```

9.9 HMILevel

Specifies current level of the HMI. An HMI level indicates the degree of user interaction possible through the HMI (e.g. TTS only, display only, VR, etc.). The HMI level varies for an application based on the type of display (i.e. Nav or non-Nav) and the user directing "focus" to other applications (e.g. phone, other mobile applications, etc.).

9.9.1 Parameter List

Name	Description	AppLink Ver. Available
FULL	The application has full use of the SYNC HMI. The app may output via TTS, display, or streaming audio and may gather input via VR, Menu, and button presses.	AppLink 1.0
LIMITED	This HMI Level is only defined for a media application using an HMI with an 8 inch touchscreen (Nav) system. The application's Show text is displayed and it receives button presses from media-oriented buttons (SEEKRIGHT, SEEKLEFT, TUNEUP, TUNEDOWN, PRESET_0-9).	AppLink 1.0
BACKGROUND	App cannot interact with user via TTS, VR, Display or Button Presses. App can perform the following operations: • Operation *AddCommand* • Operation DeleteCommand • Operation *AddSubMenu* • Operation DeleteSubMenu • Operation *CreateInteractionChoiceSet*	AppLink 1.0

	• Operation DeleteInteractionChoiceSet • Operation *SubscribeButton* • Operation UnsubscribeButton • Operation Show • Operation *UnregisterAppInterface* • Operation ResetGlobalProperties • Operation *SetGlobalProperties*	
NONE	Application has been discovered by SYNC, but application cannot send any requests or receive any notifications. An HMILevel of NONE can also mean that the user has exited the application by saying "exit appname" or selecting "exit" from the application's menu. When this happens, the application still has an active interface registration with SYNC and all SYNC resources the application has created (e.g. Choice Sets, subscriptions, etc.) still exist. But while the HMILevel is NONE, the application cannot send any messages to SYNC, except *UnregisterAppInterface*.	AppLink 1.0

9.9.2 Example Use

```
HMILevel.FULL
```

9.10 AudioStreamingState

Describes whether or not streaming audio is currently audible to the user. Though provided in every OnHMIStatus notification, this information is only relevant for applications that declare themselves as media apps in *RegisterAppInterface*.

9.10.1 Parameter List

Name	Description	AppLink Ver. Available
AUDIBLE	Currently streaming audio, if any, is audible to user.	AppLink 1.0
NON_AUDIBLE	Currently streaming audio, if any, is not audible to user. made via VR session.	AppLink 1.0

9.10.2 Example Use

```
AudioStreamingState.AUDIBLE
```

9.11 SystemContext

Indicates whether or not a user-initiated interaction is in progress, and if so, in what mode (i.e. MENU or VR).

9.11.1 Parameter List

Name	Description	AppLink Ver. Available
MAIN	No user interaction (user-initiated or app-initiated) is in progress.	AppLink 1.0
VRSESSION	VR-oriented, user-initiated or app-initiated interaction is in-progress.	AppLink 1.0
MENU	Menu-oriented, user-initiated or app-initiated interaction is in-progress.	AppLink 1.0

9.11.2 Example Use

```
SysemContext.SYSCTXT_MENU
```

9.12 AppInterfaceUnregisteredReason

Indicates reason why app interface was unregistered. The application is being disconnected by SYNC.

9.12.1 Parameter List

Name	Description	AppLink Ver. Available
IGNITION_OFF	Vehicle ignition turned off.	AppLink 1.0
BLUETOOTH_OFF	Bluetooth was turned off, causing termination of a necessary Bluetooth connection.	AppLink 1.0
USB_DISCONNECTED	USB was disconnected, causing termination of a necessary iAP connection.	AppLink 1.0
REQUEST_WHILE_IN_NONE_HMI_LEVEL	Application attempted AppLink RPC request while HMILevel=NONE. App must have HMILevel other than NONE to issue RPC requests or get notifications or RPC responses.	AppLink 1.0
TOO_MANY_REQUESTS	Either too many -- or too many per unit of time -- requests were made by the application.	AppLink 1.0
DRIVER_DISTRACTION_VIOLATION	The application has issued requests which cause driver distraction rules to be violated.	AppLink 1.0
LANGUAGE_CHANGE	The user has changed the language in effect on the SYNC platform to a language that is incompatible with the language declared by the application in its *RegisterAppInterface* request.	AppLink 1.0
MASTER_RESET	The user performed a MASTER RESET on the SYNC platform, causing removal of a necessary Bluetooth pairing.	AppLink 1.0
FACTORY_DEFAULTS	The user restored settings to FACTORY DEFAULTS on the SYNC platform.	AppLink 1.0

9.12.2 Example Use

```
AppInterfaceUnregisteredReason.IGNITION_OFF
```

9.13 HMIZoneCapabilities

Specifies HMI Zones in the vehicle.

9.13.1 Parameter List

Name	Description	AppLink Ver. Available
FRONT	Indicates HMI available for front seat passengers.	AppLink 1.0
BACK	Indicates HMI available for rear seat passengers.	AppLink 1.0

9.13.2 Example Use

```
HmiZoneCapabilities.FRONT
```

9.14 SpeechCapabilities

Contains information about TTS capabilities on the SYNC platform.

9.14.1 Parameter List

Name	Description	AppLink Ver. Available

TEXT	The SYNC platform can speak text phrases.	AppLink 1.0
SAPI_PHONEMES	The SYNC platform can speak SAPI phonemes. **Symbol Example** — **PhonemeID** - syllable boundary (hyphen) — 1 ! Sentence terminator (exclamation mark) — 2 & word boundary — 3 , Sentence terminator (comma) — 4 . Sentence terminator (period) — 5 ? Sentence terminator (question mark) — 6 _ Silence (underscore) — 7 1 Primary stress 8 — 8 2 Secondary stress 9 — 9 aa father — 10 ae cat — 11 ah cut — 12 ao dog — 13 aw foul — 14 ax ago — 15 ay bite — 16 b big — 17 ch chin — 18 d dig — 19 dh then — 20 eh pet — 21 er fur — 22 ey ate — 23 f fork — 24 g gut — 25 h help — 26 ih fill — 27 iy feel — 28 jh joy — 29 k cut — 30 l lid — 31 m mat — 32 n no — 33 ng sing — 34 ow go — 35 oy toy — 36 p put — 37 r red — 38 s sit — 39 sh she — 40	AppLink 1.0

	t talk 41 th thin 42 uh book 43 uw too 44 v vat 45 w with 46 y yard 47 z zap 48 zh pleasure 49	
LHPLUS_PHONEMES	The SYNC platform can interpret and speak LHPLUS phonemes.	AppLink 1.0
PRE_RECORDED	The SYNC platform can play pre-recorded sounds as part of a TTS operation (e.g. *Speak*, *Alert*, *PerformInteraction*, etc.). These pre-recorded sounds can be: • HELP_JINGLE • INITIAL_JINGLE • LISTEN_JINGLE • NEGATIVE_JINGLE • POSITIVE_JINGLE	AppLink 1.0
SILENCE	The SYNC platform can play the prerecorded sound of 1 second of silence (i.e. no sound at all -- like the Simon & Garfunkle song).	AppLink 1.0

9.14.2 Example Use

```
SpeechCapabilities.PRE_RECORDED
```

9.15 VRCapabilities

The VR capabilities of the connected SYNC platform.

9.15.1 Parameter List

Name	Description	AppLink Ver. Available
TEXT	The SYNC platform is capable of recognizing spoken text in the current language.	AppLink 1.0

9.15.2 Example Use

```
VrCapabilities.Text
```

9.16 ButtonName

Defines logical buttons which, on a given SYNC unit, would correspond to either physical or soft (touchscreen) buttons. These logical buttons present a standard functional abstraction which the developer can rely upon, independent of the SYNC unit. For example, the developer can rely upon the OK button having the same meaning to the user across SYNC platforms.

The preset buttons (0-9) can typically be interpreted by the application as corresponding to some user-configured choices, though the application is free to interpret these button presses as it sees fit.

The application can discover which buttons a given SYNC unit implements by interrogating the ButtonCapabilities parameter of the *RegisterAppInterface* response.

9.16.1 Parameter List

Name	Description	AppLink Ver. Available
OK	Represents the button usually labeled "OK". A typical use of this button is for the user to press it to make a selection.	AppLink 1.0
SEEKLEFT	Represents the seek-left button. A typical use of this button is for the user to scroll to the left through menu choices one menu item per press.	AppLink 1.0
SEEKRIGHT	Represents the seek-right button. A typical use of this button is for the user to scroll to the right through menu choices one menu item per press.	AppLink 1.0
TUNEUP	Represents a turn of the tuner knob in the clockwise direction one tick.	AppLink 1.0
TUNEDOWN	Represents a turn of the tuner knob in the counter-clockwise direction one tick.	AppLink 1.0
PRESET_0	Represents the preset 0 button.	AppLink 1.0
PRESET_1	Represents the preset 1 button.	AppLink 1.0
PRESET_2	Represents the preset 2 button.	AppLink 1.0
PRESET_3	Represents the preset 3 button.	AppLink 1.0
PRESET_4	Represents the preset 4 button.	AppLink 1.0
PRESET_5	Represents the preset 5 button.	AppLink 1.0
PRESET_6	Represents the preset 6 button.	AppLink 1.0
PRESET_7	Represents the preset 7 button.	AppLink 1.0
PRESET_8	Represents the preset 8 button.	AppLink 1.0
PRESET_9	Represents the preset 9 button.	AppLink 1.0

9.16.2 Example Use

```
ButtonName.OK
```

9.17 MediaClockFormat

Indicates the format of the time displayed on the connected SYNC unit.

9.17.1 Parameter List

Name	Description	AppLink Ver. Available
CLOCK1	• maxHours = 19 • maxMinutes = 59 • maxSeconds = 59	AppLink 1.0
CLOCK2	• maxHours = 59 • maxMinutes = 59 • maxSeconds = 59\	AppLink 1.0
CLOCKTEXT1	• 5 characters possible • Format: 1|sp c :|sp c c • 1|sp : digit "1" or space • c : character out of following character set: sp|0-9|[letters, see TypeII column in XLS. • :|sp : colon or space • used for Type II headunit	AppLink 1.0
CLOCKTEXT2	• 5 characters possible • Format: 1|sp c :|sp c c	AppLink 1.0

	• 1|sp : digit "1" or space • c : character out of following character set: sp|0-9|[letters, see CID column in XLS. • :|sp : colon or space • used for CID headunit difference between CLOCKTEXT1 and CLOCKTEXT2 is the supported character set	
CLOCKTEXT3	• 6 chars possible • Format: 1|sp c c :|sp c c • 1|sp : digit "1" or space • c : character out of following character set: sp|0-9|[letters, see Type 5 column in XLS]. • :|sp : colon or space • used for Type V headunit difference between CLOCKTEXT1 and CLOCKTEXT2 is the supported character set	AppLink 1.0

9.17.2 Example Use

```
MediaClockFormat.CLOCK1
```

9.18 DisplayType

Identifies the various display types used by SYNC. See AppLink TDK and Head Unit Guide for further information regarding the displays.

9.18.1 Parameter List

Name	Description	AppLink Ver. Available
CID	This display type provides a 2-line x 20 character "dot matrix" display.	AppLink 1.0
TYPE2		AppLink 1.0
TYPE5		AppLink 1.0
NGN	This display type provides an 8 inch touchscreen display.	AppLink 1.0
GEN2_4_DMA		AppLink 1.0
GEN2_8_DMA		AppLink 1.0
GEN2_8_HUD		AppLink 1.0

9.18.2 Example Use

```
DisplayType.NGN
```

9.19 CharacterSet

Character sets supported by SYNC.

9.19.1 Parameter List

Name	Description	AppLink Ver. Available
TYPESET2		AppLink 1.0
TYPESET5		AppLink 1.0
CIDSET1		AppLink 1.0

CIDSET2		AppLink 1.0

9.19.2 Example Use

```
CharacterSet.CID1SET
```

9.20 TextFieldName

Names of the text fields that can appear on a SYNC display.

9.20.1 Parameter List

Name	Description	AppLink Ver. Available
mainField1	The top line of the persistent display. Applies to Show.	AppLink 1.0
mainField2	The bottom line of the persistent display. Applies to Show.	AppLink 1.0
statusBar	The status bar on the NGN display. Applies to Show.	AppLink 1.0
mediaClock	Text value for MediaClock field. Must be properly formatted according to MediaClockFormat. Applies to Show. This field is commonly used to show elapsed or remaining time in an audio track or audio capture	AppLink 1.0
mediaTrack	The track field of NGN type ACMs. This field is only available for media applications on a NGN display. Applies to Show. This field is commonly used to show the current track number	AppLink 1.0
alertText1	The top line of the persistent display. Applies to *Alert*.	AppLink 1.0
alertText2	The bottom line of the persistent display. Applies to *Alert*.	AppLink 1.0

9.20.2 Example Use

```
TextFieldName.mainField1
```

9.21 TextAlignment

The list of possible alignments of text in a field.

9.21.1 Parameter List

Name	Description	AppLink Ver. Available
LEFT_ALIGNED	Text aligned left.	AppLink 1.0
RIGHT_ALIGNED	Text aligned right.	AppLink 1.0
CENTERED	Text aligned centered.	AppLink 1.0

9.21.2 Example Use

```
TextAlignment.CENTERED
```

9.22 GlobalProperty

Properties of a user-initiated VR interaction (i.e. interactions started by the user pressing the PTT button).

9.22.1 Parameter List

Name	Description	AppLink Ver. Available
HELPPROMPT	The help prompt to be spoken if the user needs assistance during a user-initiated interaction.	AppLink 1.0
TIMEOUTPROMPT	The prompt to be spoken if the user-initiated interaction times out waiting for the user's verbal input.	AppLink 1.0

9.22.2 Example Use

```
GlobalProperty.HELPPROMPT
```

9.23 TBTState

Describes possible states of turn-by-turn module.

9.23.1 Parameter List

Name	Description	AppLink Ver. Available
ROUTE_ACCEPTED		AppLink 1.0
ROUTE_REFUSED		AppLink 1.0
ROUTE_CANCELLED		AppLink 1.0
ROUTE_UPDATE_REQUEST	Indicates that driver requested a route update.	AppLink 1.0

9.23.2 Example Use

```
TBTState.ROUTE_ACCEPTED
```

9.24 DriverDistractionState

Describes possible states of driver distraction.

9.24.1 Parameter List

Name	Description	AppLink Ver. Available
DD_ON	Driver distraction rules are in effect.	AppLink 1.0
DD_OFF	Driver distraction rules are NOT in effect.	AppLink 1.0

9.24.2 Example Use

```
DriverDistractionState.DD_ON
```

10 Connected and In-Focus Applications

A mobile application which is running in focus should be able to write to the display. Examples of persistent data may include:

- Writing metadata information for streaming audio
- Displaying a one or two line informational message
- Writing to the media timer

The mobile application itself is responsible for writing data and displaying information to the user. Information regarding the type of display and its capabilities shall be made available to the mobile application, and it shall be the responsibility of the mobile application to adapt its messages to the vehicle display. A message or messages shall be sent by SYNC® to indicate to the mobile application that is has control or is updating the display.

10.1 Media Applications

10.1.1 Main Screen

Diagrams to be added

10.1.1.1 Mainfields 1 – 2

Dynamic text fields that can be written by the application to inform the user of important information through the Show RPC.

10.1.1.2 Media Track

Dynamic field usually containing the application name that can be written through the Show RPC.

Note: This feature is only available on the NGN.

10.1.1.3 Media Clock

System generated clock for use on media timers called through the *SetMediaClockTimer* RPC. System will allow application count up, count down, pause, resume and clear functions.

10.1.1.4 On Screen Buttons

For NGN only systems the first 4 AddCommands will be displayed at buttons on the touchscreen. See diagram AppLink TDK and Head Unit Guide for more details.

10.2 Non-Media Applications

10.2.1 Main Screen

Diagrams to be added

10.2.1.1 Mainfields 1 – 2

Dynamic text fields that can be written by the application to inform the user of important information through the Show RPC.

Note: NGN units will only have Mainfield1 shown on the screen for Non-media application types

10.2.1.2 On Screen Buttons

For NGN only systems the first 4 AddCommands will be displayed at buttons on the touchscreen. See diagram AppLink TDK and Head Unit Guide for more details.