RUN3TV™

The NEXTGEN TV Web Platform

Run3TV Implementation Guidelines: I-Frame Sub-Application Integration

Version:	3.0
Date:	May 29th 2024
Doc:	R3TV-IG-0212

Framework version: 2.3

Revision history

Version	Date	Framework Version	Update
v1.0 (Final)	February 20th 2024	2.3	First Release
v2.0 (Final)	May 2nd 2024	2.3	Added missing AMP functions and events. Added the following functions: ● aeat() ● alerting() ● tagReplace() Added the following events: ● onAlert()
v3.0 (Final)	**May 29th 2024**	**2.3**	Added the following event: ● onActiveAlerts()

Copyright notice

Contents

1. Glossary

Glossary of unfamiliar words and acronyms.

Term	Definition
AMP	Application Media Player (ATSC 3.0)
RMP	Receiver Media Player (ATSC 3.0)

2. References

ID	Publisher	Document
R3TV-IG-0231	Run3TV	Run3TV Implementation Guidelines: RSS Feed Specification
R3TV-IG-0512	Run3TV	Run3TV Implementation Guidelines: SIMID Creatives
R3TV-IG-0203	Run3TV	Run3TV Implementation Guidelines: Run3TV SDK APIs
R3TV-IG-0204	Run3TV	Run3TV Implementation Guidelines: Run3TV Application Management
A/344	ATSC	ATSC Standard: “ATSC 3.0 Interactive Content” *(The applicable version(s) of this document are dependent upon the Sub-App markets’ receiver base)*

3. Introduction

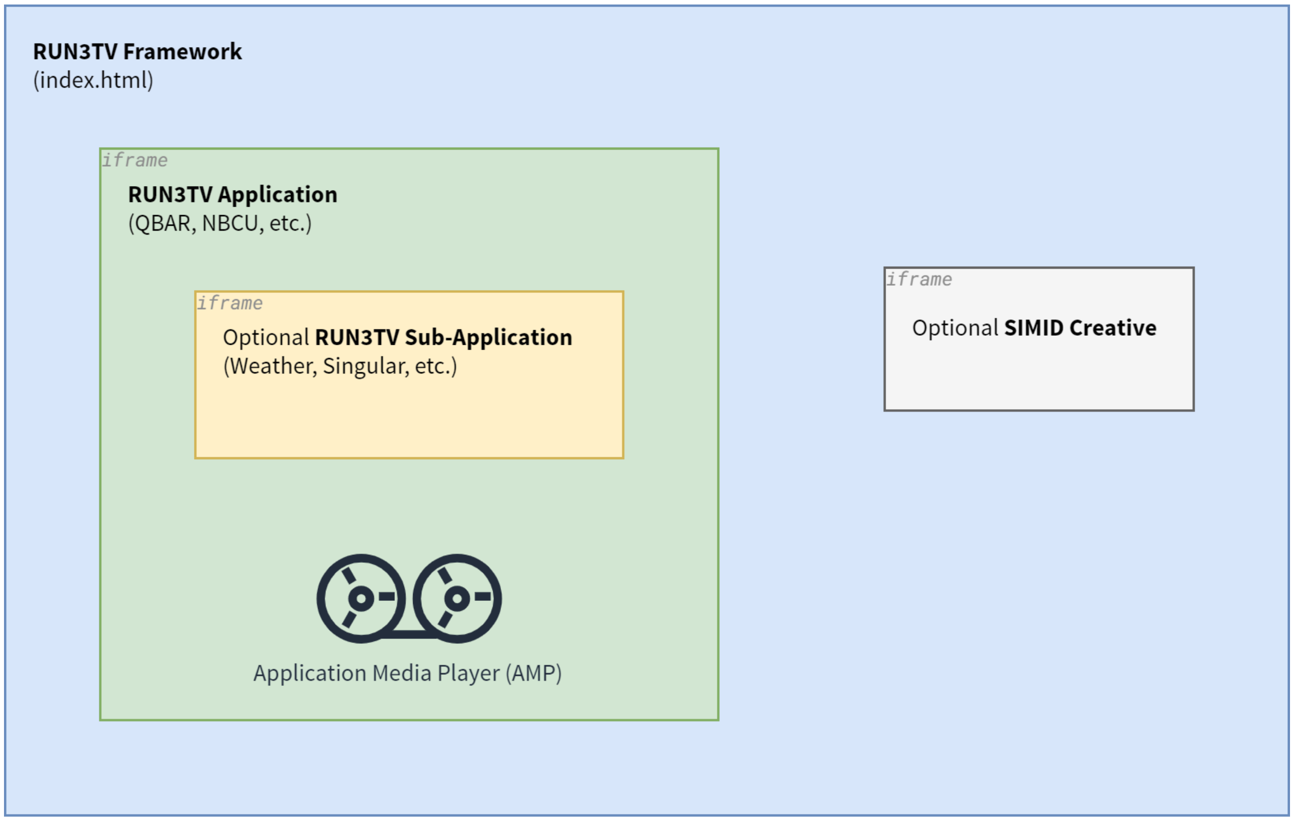
The Run3TV Framework typically consists of at least two pages, the top (static) Framework and the dynamic (iframe) Application page as a child.

The Framework can switch in and out Run3TV Applications within the iframe.

Optionally, the Run3TV Application can itself include **Run3TV Sub-Applications** (Sub-Apps) in an iframe. These Sub-Apps provide additional interactivity within the main Run3TV Application.

Note Additionally, SIMID Creative applications can be launched through VAST signaling. SIMID Creatives share a similar interface to Sub-Apps. For more details see [R3TV-IG-0512].

This hierarchy is summarized in the diagram below.



For example: Within the Q-Bar application, it is possible to define rails (using RSS feeds[1]) that contain items that refer to Sub-Apps. Viewers can navigate to these items and access additional functionality.

When implemented correctly, Sub-Apps form a seamless viewer experience with the Run3TV (main) Application.

This document describes:

- The application lifecycle of Sub-Apps
- The feature set of Sub-Apps; and,
- The API that Sub-Apps must implement.

This document refers to Run3TV Applications as Run3TV (Main) Applications, to lower the possibility of confusing Run3TV Sub-Apps with Run3TV Applications.

The descriptions throughout use the Q-Bar application as the example Run3TV (Main) Application. Different applications may provide access to Sub-Apps in alternate ways, however the underlying lifecycle remains the same.

3.1. Example Sub-Apps In The Starter Kit

The Run3TV Starter Kit provides the following example Sub-Apps:

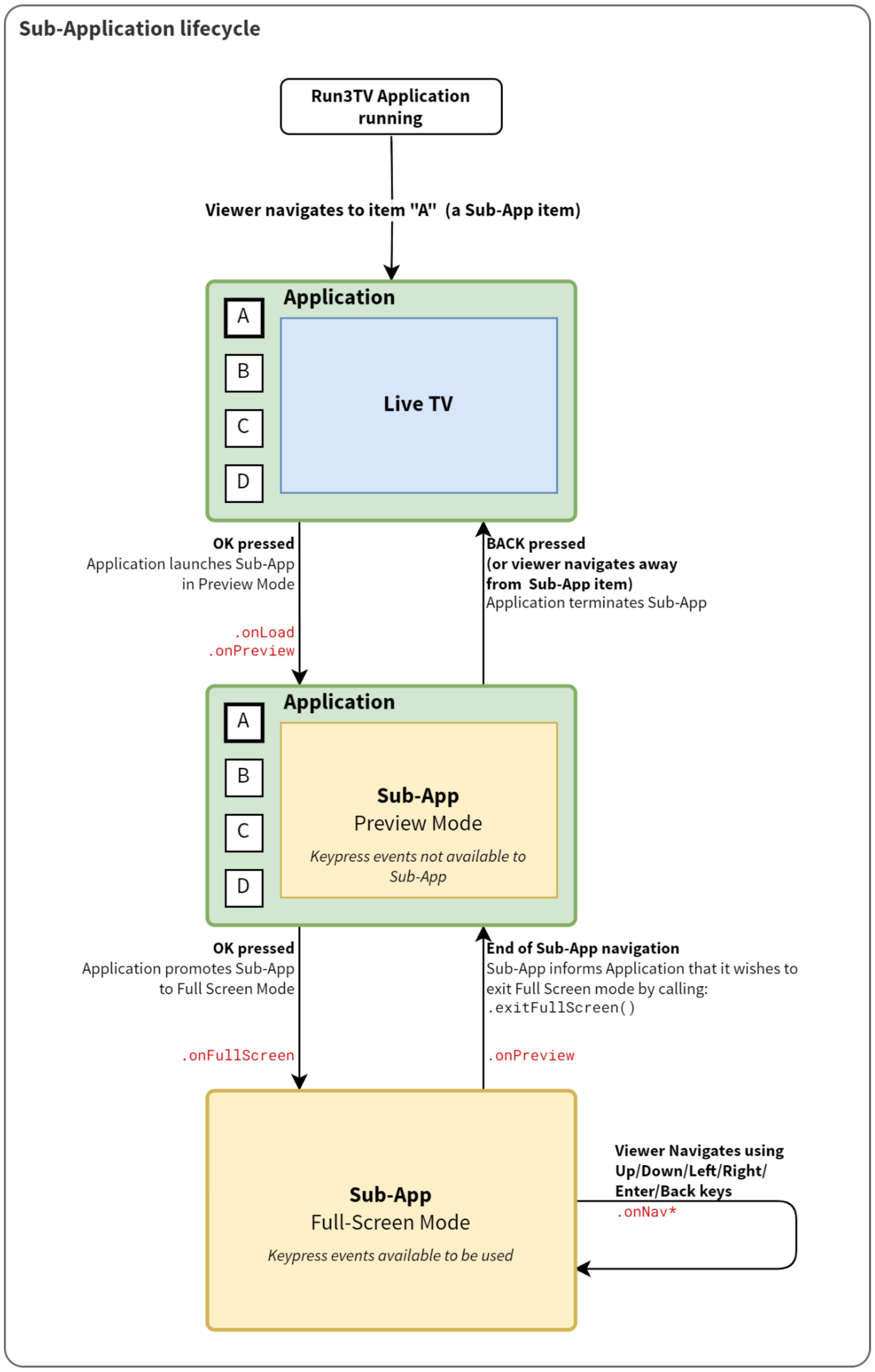
Starter Kit Location	**Description**
`public/london/apps/ run3tv-sub-app-basic`	A simple Sub-App that presents a static weather image in Preview Mode and Full Screen mode. It demonstrates how to follow the Sub-App Application Lifecycle
`public/london/apps/ run3tv-sub-app-video`	A simple Sub-App that builds on the basic Sub-App above to present video
`public/london/apps/ run3tv-sub-app-navigation`	A simple Sub-App that builds on the basic Sub-App above to introduce navigation control
`public/london/apps/ run3tv-sub-app-video-control`	A simple Sub-App that builds on the video Sub-App above to allow video playback control

These applications provide complete, worked examples of Sub-Apps, and are described in more detail in Annex A - Example Sub-Apps.

[1] For details on how to create RSS feeds, see [R3TV-IG-0231].

4. Sub-App Application Lifecycle

This section describes the application lifecycle of Run3TV Sub-Apps. The diagram below provides a high-level overview of this lifecycle. The red function calls refer to events within the `subApp.run3tv.min.js` api :


Sub-Application lifecycle
Run3TV Application running
Viewer navigates to item "A" (a Sub-App item)
Application
A
B
C
D
Live TV
OK pressed
Application launches Sub-App in Preview Mode
.onLoad
.onPreview
BACK pressed (or viewer navigates away from Sub-App item)
Application terminates Sub-App
Application
A
B
C
D
Sub-App
Preview Mode
Keypress events not available to Sub-App
OK pressed
Application promotes Sub-App to Full Screen Mode
.onFullScreen
End of Sub-App navigation
Sub-App informs Application that it wishes to exit Full Screen mode by calling:
.exitFullScreen()
.onPreview
Sub-App
Full-Screen Mode
Keypress events available to be used
Viewer Navigates using Up/Down/Left/Right/Enter/Back keys
.onNav*

Sub-Apps can be presented either of two modes:

- **Preview Mode** - Presentation is scaled down (with the same 16:9 aspect ratio as full screen content) to fit within the frame of the Run3TV (Main) Application. No key events are passed on to the Sub-App
- **Full Screen Mode** - Presentation is scaled to fill the screen. Sub-Apps receive events for certain keys from the Run3TV (Main) Application.

Run3TV (Main) Applications and Sub-Apps inform each other of lifecycle changes via the `subApp.run3tv.min.js` events interface. In addition, when the Sub-App is in Full Screen Mode, the same interface is used to:

- Enable the Run3TV (Main) Application to inform the Sub-App of certain key events
- Enable the Sub-App to to control and monitor A/V playback using the Run3TV (Main) Application's Application Media Player (AMP)

4.1. Sub-App Modes And subApp.run3tv.min.js Events And Functions

The table below summarizes which `subApp.run3tv.min.js` functions and events are applicable for each Sub-App Mode. The `subApp.run3tv.min.js` interface is described in detail in Interface API For subApp.run3tv.min.js.

Run3TV provides a local version of `subApp.run3tv.min.js` for development. This must not be used in the production environment.

Please contact your Run3TV representative to gain access to the hosted production version of this API.

	`subApp.run3tv.min.js` Events & Functions	**Description**	**Defined by**	**Applicable in**	
				Preview Mode	**Full Screen Mode**
Lifecycle	`.onload()`	Fired when the Sub-App is first loaded	This document	✔	—
	`.onPreview()`	Fired when the Run3TV (Main) Application moves the Sub-App to Preview Mode	This document	✔ *(on first start)*	✔
	`.onFullScreen()`	Fired when the Run3TV (Main) Application moves the Sub-App to Full Screen Mode	This document	✔	—
	`.exitFullScreen()`	The Sub-App must call this function to inform the Run3TV (Main) Application to exit Full Screen Mode	This document	—	✔
	`.isFullScreen()`	Returns `true` if the Sub-App is in Full Screen Mode, otherwise returns `false`	This document	✔	✔
	`.getPreviewWindowScale()`	Returns the position and scale factor of the Preview Window	This document	✔	✔

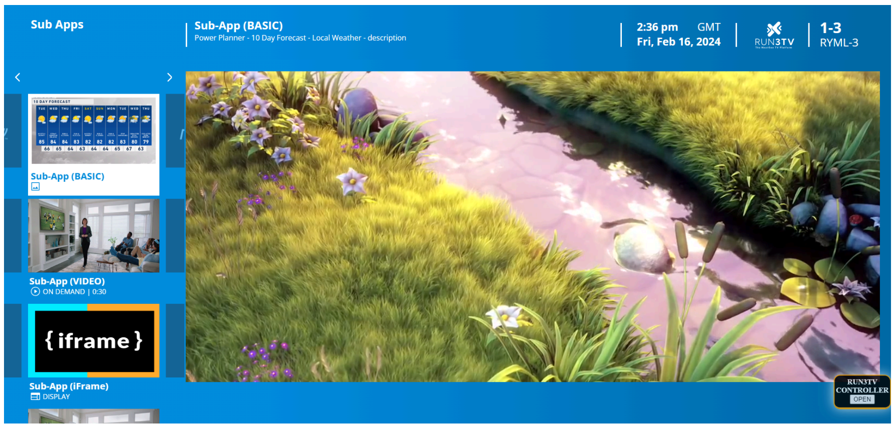
	`subApp.run3tv.min.js` **Events & Functions**	**Description**	**Defined by**	**Applicable in**	
				Preview Mode	**Full Screen Mode**
Navigation	`.onNavNext()`	Fired when the viewer presses **RIGHT**	This document	—	✔
	`.onNavBack()`	Fired when the viewer presses **LEFT**	This document	—	✔
	`.onNavUp()`	Fired when the viewer presses **UP**	This document	—	✔
	`.onNavDown()`	Fired when the viewer presses **DOWN**	This document	—	✔
	`.onNavEnter()`	Fired when the viewer presses **OK** or **SELECT**	This document	—	✔
	`.onNavExit()`	Fired when the viewer presses **BACK**	This document	—	✔
Media Control	`.ampAddtrack()`	Adds a subtitle track to the AMP	[R3TV-IG-0203]	✔	✔
	`.ampDashLicense()`	An event, fired when the AMP requests a DASH license	[R3TV-IG-0203]	✔	✔
	`.ampDuration()`	The duration of the AMP content	[R3TV-IG-0203]	✔	✔
	`.ampGetCurrentTime()`	Gets the current playback time of the AMP	[R3TV-IG-0203]	✔	✔
	`.ampGetposter()`	Gets the “Loading” image	[R3TV-IG-0203]	✔	✔
	`.ampHidetracks()`	Hides one or more subtitle tracks	[R3TV-IG-0203]	✔	✔
	`.ampIsLoop()`	Returns the AMP loop state	[R3TV-IG-0203]	✔	✔
	`.ampLoop()`	Enables or disables looping of the video	[R3TV-IG-0203]	✔	✔
	`.ampMediaTimeChange()`	An event, fired at regular intervals during AMP playback	[R3TV-IG-0203]	✔	✔
	`.ampMuted()`	Returns the AMP muted state	[R3TV-IG-0203]	✔	✔
	`.ampPause()`	Pauses the AMP	[R3TV-IG-0203]	✔	✔
	`.ampPlay()`	Starts playback. (Not required if the video to play is configured in the RSS feed)	[R3TV-IG-0203]	✔	✔
	`.ampPlaybackStateChange()`	An event, fired when the AMP’s state changes	[R3TV-IG-0203]	✔	✔
	`.ampRemovetrack()`	Removes one or all subtitle tracks	[R3TV-IG-0203]	✔	✔
	`.ampScale()`	Sets the scale and position of the AMP	[R3TV-IG-0203]	✔	✔

	`subApp.run3tv.min.js` **Events & Functions**	**Description**	**Defined by**	**Applicable in**	
				Preview Mode	**Full Screen Mode**
	`.ampSetCurrentTime()`	Sets the playback time of the AMP	[R3TV-IG-0203]	✔	✔
	`.ampSetposter()`	Sets the "Loading" image	[R3TV-IG-0203]	✔	✔
	`.ampShowtracks()`	Presents a subtitle track	[R3TV-IG-0203]	✔	✔
	`.ampStart()`	Prepares the AMP for video playback. (Not required if the video to play is configured in the RSS feed)	[R3TV-IG-0203]	✔	✔
	`.ampState()`	Returns the current state of the AMP	[R3TV-IG-0203]	✔	✔
	`.ampStop()`	Stops playback	[R3TV-IG-0203]	✔	✔
	`.ampTime()`	The current playback time of the AMP	[R3TV-IG-0203]	✔	✔
	`.ampTracks()`	Returns the total number of subtitle tracks	[R3TV-IG-0203]	✔	✔
	`.ampType()`	Sets the type of content to play in the AMP	[R3TV-IG-0203]	✔	✔
	`.ampUrl()`	The URL of the AMP content	[R3TV-IG-0203]	✔	✔
Alerting	`.aeat()`	Returns the current AEAT (Advanced Emergency Alert Table) XML data	[R3TV-IG-0203]	✔	✔
	`.alerting()`	the various alerting metadata from the current ATSC 3.0 broadcast	[R3TV-IG-0203]	✔	✔
	`.onAlert()`	Fired when a new alert (that the viewer has subscribed to is received). Provides all known alerts	[R3TV-IG-0203]	✔	✔
	`.onActiveAlerts()`	Fired when an alert (that the viewer has subscribed to is received) becomes active. Provides all currently active alerts	[R3TV-IG-0203]	✔	✔
Misc.	`.tagReplace()`	takes a string and substitutes any Replaceable Field Tags (as detailed in [R3TV-IG-0204]) with their replacement values	[R3TV-IG-0203]	✔	✔

4.2. Prior To Launch

Viewers can navigate within the Run3TV (Main) Application to highlight content items such as VOD assets, images or Sub-Apps. When highlighted (but not selected), the item thumbnails for these content items are presented — the underlying content is not displayed.

An example of a Sub-App item that has been highlighted but not selected is shown below. The “Basic” item is highlighted and live TV is rendering.



Sub-Apps are not launched until the viewer selects them from the on-screen menu by pressing **OK** or **SELECT**.

4.3. Launching And Preview Mode

Run3TV Sub-Apps are launched by the Run3TV (Main) Application when the viewer presses **OK** or **SELECT** on the content item.

Sub-Apps are first launched into **Preview Mode**. This mode places the contents of the Sub-App within the existing frame of the Run3TV (Main) Application's scaled video window.

Therefore, the Run3TV (Main) Application will fire two events from the `subApp.run3tv.min.js` interface when first launching:

- `.onLoad()`
- `.onPreview()`

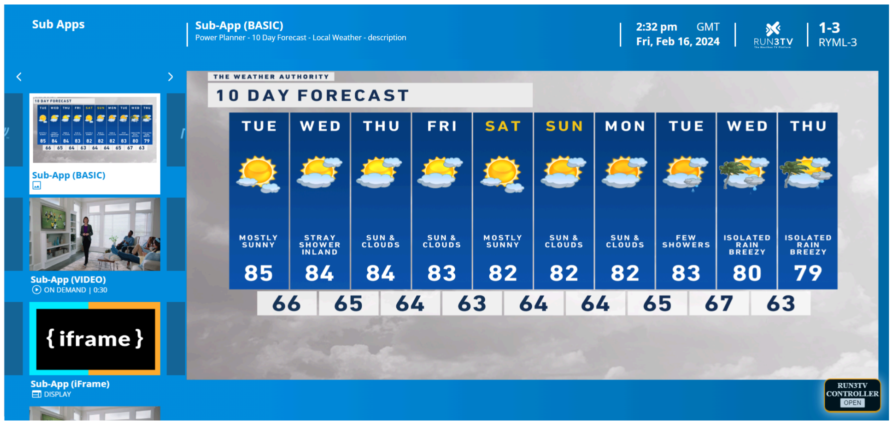
When a Sub-App is first launched, the Run3TV (Main) Application stops the Receiver Media Player (RMP).

Note All example Sub-Apps provided in the Starter Kit follow the application lifecycle detailed here. See run3tv-sub-app-basic for a minimal example.

When in Preview Mode, the Sub-App is presented in the same aspect ratio (16:9) as when in Full Screen Mode.

For details on how to determine the presented size of the Sub-App, see Window size, position and scaling in Preview Mode.

The screenshot below shows the run3tv-sub-app-basic Sub-App in Preview Mode within the Q-Bar.



THE WEATHER AUTHORITY

10 DAY FORECAST

TUE	WED	THU	FRI	SAT	SUN	MON	TUE	WED	THU
MOSTLY SUNNY	STRAY SHOWER INLAND	SUN & CLOUDS	SUN & CLOUDS	MOSTLY SUNNY	SUN & CLOUDS	SUN & CLOUDS	FEW SHOWERS	ISOLATED RAIN BREEZY	ISOLATED RAIN BREEZY
85	84	84	83	82	82	82	83	80	79

66 65 64 63 64 64 65 67 63

Whilst in Preview Mode, Sub-Apps will **not** receive navigation key events from the `subApp.run3tv.min.js` interface. Therefore, Sub-Apps in Preview Mode shall not present navigation buttons (other than a call to action) or on-screen media controls.

The Run3TV (Main) Application will terminate the Sub-App (and restart the RMP) when the viewer chooses to navigate away from the Sub-App's content item.

Note For details on configuring Sub-Apps to include video, see Including video within a Sub-App.

4.4. Full Screen Mode

When the viewer clicks **OK** a second time within the Run3TV (Main) Application to open a Sub-App, the Run3TV (Main) Application will promote the Sub-App from Preview Mode to Full Screen Mode.

The Run3TV (Main) Application informs the Sub-App of the promotion to Full Screen Mode by firing the `.onFullScreen()` event from `subApp.run3tv.min.js`.

Whilst in Full Screen Mode, the following key events will be passed to the Sub-App via the `subApp.run3tv.min.js` interface:

Key	**`subApp.run3tv.min.js` event**
Right	`.onNavNext()`
Left	`.onNavBack()`
Up	`.onNavUp()`
Down	`.onNavDown()`
OK or **SELECT**	`.onNavEnter()`
BACK	`.onNavExit()` *Sub-Apps must always capture this event and call `.exitFullScreen()` as appropriate*

Note Sub-Apps do not receive focus, so cannot capture key events directly.

The Run3TV (Main) Application will inform the Sub-App to revert from Full Screen Mode to Preview Mode by sending the `.onPreview()` event from `subApp.run3tv.min.js`.

Note The run3tv-sub-app-navigation example Sub App (available in the Starter Kit) provides a worked example of how to make use of navigation events provided by `subApp.run3tv.min.js`.

The Sub-App is responsible for exiting Full Screen mode. To do so, it **must** call `.exitFullScreen()` when the viewer wishes to return back to the Run3TV (Main) Application. This must occur when the viewer presses the **BACK** key to complete navigation within the Sub-App. In other words, all Sub-Apps must always register for `.onNavExit()` events and call `.exitFullScreen()` as appropriate.

Depending on the intended user experience of the Sub-App, it may be appropriate to call `.exitFullScreen()` at other times.

The Sub-App must remove any navigation controls and on-screen media playback controls when it returns to Preview Mode.

Please refer to Sub-App Development Guidelines for more details on viewer navigation within Sub-Apps.

5. Associating A Sub-App To A Content Rail

This section describes how to associate a Sub-App to a content rail within the Q-Bar application.

This is performed by configuring the RSS feed to include a reference to a `<media:content>` object with a `medium` attribute set to be “`document`”.

For more information on how to define RSS feeds in Run3TV applications (including examples in JSON format), see [R3TV-IG-0231].

5.1. Sub-App Minimal Example

An example RSS item object for a simple Sub-App (such as those provided in the Starter Kit) is provided below.

RSS example 1: A simple Sub-App

```
1    <item>
2      <title>Sub-App with Video</title>
3      <description>A Sub-App with a video asset</description>
4      <media:thumbnail url="/london/Station-3/q-bar/img/thumb.png"
5                       width="300" height="169"/>
6
7      <media:group>
8        <media:content medium="document" type="text/html"
9             url="/london/apps/run3tv-sub-app-basic/index.html"/>
10     </media:group>
11
12   </item>
```

In this example, the `<media:content>` object (line 8 & 9) informs the Run3TV (Main) Application that the item to present is a Sub-App.

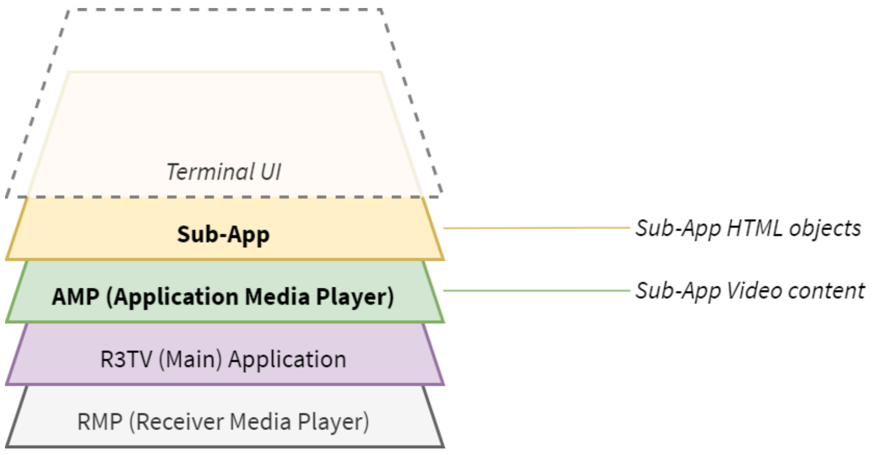
5.2. Sub-Apps With Video

Sub-Apps can include audio/video content. This audio/video will be rendered by the Run3TV (Main) Application's AMP (Application Media Player) whilst the Sub-App is in both Preview Mode and Full Screen Mode.

When switching between modes, the Run3TV (Main) Application will continue to play the audio/video.

The Run3TV (Main) Application will stop AMP media playback (and restore RMP playback) when it terminates the Sub-App. In other words, Sub-Apps do not need to stop AMP playback on termination.

The Z-order of the various applications and media players is summarized below. This order cannot be modified by the Sub-App.



AMP video will only be visible if the Sub-App provides "open" space to allow the video to cut through the Sub-App's own Z-Order layer.

It is the responsibility of the Sub-App to present any required video player controls.

The content to be played can be configured in one of two ways:

RSS-Configured Video Startup:

The video can be defined within the `<media:group>` of the `<item>` in the RSS file. When the R3TV (Main) Application first launches the Sub-App, it also starts video playback in the AMP

Programmatic Video Startup:

The Sub-App is responsible for starting video playback in the AMP

With either method of video startup, the Sub-App can control video playback using the `.amp*` functions and events of the `subApp.run3tv.min.js` interface.

These two configuration options are described in the following sections.

5.2.1. RSS-Configured Video Startup

It is possible to configure the Q-Bar (or equivalent) RSS feed to play video alongside the Sub-App by defining a `<media:group>` object that contains both Sub-App and video `<media:content>` elements. This is useful when the Sub-App requires a video that is known in advance of the Sub-App launching.

Video configured this way will start playback as soon as the Sub-App first enters Preview Mode. The Run3TV (Main) Application will stop any AMP video playback when the Sub-App exits.

⚠ When using this method to play video, the Sub-App must be defined in the `<media:group>` element **before** the video.

An example RSS `<item>` for such a Sub-App is shown below.

RSS example 2: Sub-App plus video rendered directly by the Run3TV (Main) Application

```
1    <item>
2      <title>Sub-App with Video</title>
3      <description>A Sub-App with a video asset</description>
4      <media:thumbnail url="/london/Station-3/q-bar/img/thumb.png"
5                       width="300" height="169"/>
6
7      <media:group>
8
9        <media:content medium="document" type="text/html" url="/london/apps/nextgen/en.html"/>
10
11       <media:content medium="video" type="application/vnd.apple.mpegurl" expression="loop"
12                      url="https://example.com/video.m3u8" duration="67.28"/>
13
14       <media:embed url="run3tv://player/AMP">
15         <media:param name="loopPlayback">true</media:param>
16       </media:embed>
17
18     </media:group>
19   </item>
```

In the example above:

- **Line 9** defines the Sub-App to launch.
  The Sub-App must always be defined before the video.
- **Lines 11 & 12** define the video for the Run3TV (Main) Application to present whilst the Sub-App is in Preview Mode and Full Screen Mode
- **Lines 14 - 16** define additional configuration for the video

Note The Run3TV (Main) Application's on-screen media playback controls are not available within Sub-Apps. Sub-Apps must present their own playback controls if required.

The `<media:embed>/<media:param>` parameter `showPlaybackControls` within the RSS feed has no effect for content items containing Sub-Apps.

For more detailed information on the Run3TV RSS profiling, see [R3TV-IG-0231].

5.2.2. Programmatic Video Startup

The Sub-App can programmatically select content to play. This is useful when the Sub-App needs to select specific content to play back at runtime.

Video playback via the AMP is performed using the `subApp.run3tv.min.js` interface.

To start video playback, the following two commands must be run:

- `amp.start()` - to prepare the AMP for playback
- `amp.play()` - to inform the AMP to start playing the video

Details of these functions are described in the next section.

The Run3TV (Main) Application will stop any AMP video playback when the Sub-App exits.

5.2.3. Video Playback Control API

Sub-Apps can start, control and stop video playback within the AMP using the video playback API available within the `subApp.run3tv.min.js` interface. This interface is detailed in Interface API For subApp.run3tv.min.js.

The run3tv-sub-app-video-control Sub-App (provided as part of the Starter Pack). provides an example of a Sub-App with on-screen video controls.

6. Sub-App Development Guidelines

This section provides a summary of the guidelines (and guidance) for Sub-App development.

Run3TV Sub-Apps are — *by design* — not intended to be fully-featured applications. If additional functionality beyond basic navigation is required, consider creating a Run3TV (Main) Application instead of a Sub-App.

6.1. Window size, position and scaling in Preview Mode

Whilst Sub-Apps are always presented in a 16:9 aspect ratio, they may not always fill the screen. Depending on the graphical design of the Sub-App, it may be appropriate to adjust how the GUI is presented depending on the window size.

The window size can be determined using the following function from the `subApp.run3tv.min.js` interface:

```
.getPreviewWindowScale()
```

This function will always return the size, position and scaling factor of the Preview Mode window, even when the Sub-App is in Full Screen Mode.

For more details, see .getPreviewWindowScale().

6.2. Navigation

Sub-Apps cannot capture keypresses directly. Instead, the navigation event API provided by the `subApp.run3tv.min.js` interface shall be used to detect keypresses.

Sub-Apps must be designed to make use of these navigation events and update their GUI as appropriate so that the viewer can easily navigate within the Sub-App.

Browser based focusable navigation is not supported within Sub-Apps.

The Sub-App user interface must be designed in such a way that the viewer can exit Full Screen Mode (and return control to the Run3TV (Main) Application) by pressing the **BACK** key. Other keys may also be used to enable the viewer to exit the Sub-App, so long as their function is made clear. Depending on the intended user experience of the Sub-App, it may be appropriate to exit Full Screen mode when a audio/video item has completed playback.

7. Sub-App Development Environment

This section will follow in a future version of this document.

8. Interface API For subApp.run3tv.min.js

This section provides a description of the `subApp.run3tv.min.js` interface API.

This API includes AMP (Application Media Player) control functions and events that match the interface described in [R3TV-IG-0203]. To avoid duplication, these are *not* defined in the current document; please refer to [R3TV-IG-0203] for details on the following events and functions:

`subApp.run3tv.min.js` **Event / Function**		**Framework SDK Event / Function**
`.ampAddtrack()`	⇔	`fmw.amp.addtrack()`
`.ampDashLicense()`	⇔	`fmw.ampDashLicense()`
`.ampDuration()`	⇔	`fmw.amp.duration()`
`.ampGetCurrentTime()`	⇔	`fmw.amp.getCurrentTime()`
`.ampGetposter()`	⇔	`fmw.amp.getposter()`
`.ampHidetracks()`	⇔	`fmw.amp.hidetracks()`
`.ampIsLoop()`	⇔	`fmw.amp.isLoop()`
`.ampLoop()`	⇔	`fmw.amp.loop()`
`.ampMediaTimeChange()`	⇔	`fmw.amp.mediaTimeChange()`
`.ampMuted()`	⇔	`fmw.amp.muted()`
`.ampPause()`	⇔	`fmw.amp.pause()`
`.ampPlay()`	⇔	`fmw.amp.play()`
`.ampPlaybackStateChange()`	⇔	`fmw.amp.playbackStateChange()`
`.ampRemovetrack()`	⇔	`fmw.amp.removetrack()`
`.ampScale()`	⇔	`fmw.amp.scale()`
`.ampSetCurrentTime()`	⇔	`fmw.amp.setCurrentTime()`
`.ampSetposter()`	⇔	`fmw.amp.setposter()`

`subApp.run3tv.min.js` **Event / Function**		**Framework SDK Event / Function**
`.ampShowtracks()`	⇔	`fmw.amp.showtracks()`
`.ampStart()`	⇔	`fmw.amp.start()`
`.ampState()`	⇔	`fmw.amp.state()`
`.ampStop()`	⇔	`fmw.amp.stop()`
`.ampTime()`	⇔	`fmw.amp.time()`
`.ampTracks()`	⇔	`fmw.amp.tracks()tagReplace`
`.ampType()`	⇔	`fmw.amp.type()`
`.ampUrl()`	⇔	`fmw.amp.url()`

The API includes the following alerting methods and events, as defined in [R3TV-IG-0203]:

`subApp.run3tv.min.js` **Event / Function**		**Framework SDK Event / Function**
`.aeat()`	⇔	`fmw.aeat()`
`.onAlert()`	⇔	`onAlert()`
`.onActiveAlerts()`	⇔	`onActiveAlerts()`
`.alerting()`	⇔	`fmw.alerting()`

The API includes the following miscellaneous methods and events, as defined in [R3TV-IG-0203]:

`subApp.run3tv.min.js` **Event / Function**		**Framework SDK Event / Function**
`.tagReplace()`	⇔	`fmw.tagReplace()`

8.1. .onload()

The `.onload()` event is fired when the Run3TV (Main) Application first launches the Sub-App.

Syntax

```
subApp.onload( functionList, // Array of Strings
               isFullScreen  // Boolean
             );
```

Parameters

`functionList <Array of String>`

A list of all `subApp.run3tv.min.js` functions and events available for use by the Sub-App

`isFullScreen <Boolean>`

If set to `true`, the Sub-App is in Full Screen Mode on launch. Otherwise set to `false`.

8.2. .onPreview()

The`.onPreview()` event is fired when the Run3TV (Main) Application moves the Sub-App into Preview Mode.

The Run3TV (Main) Application will fire this event any time it moves the Sub-App into Preview Mode. For example:

- When first launching the Sub-App into Preview Mode(after first firing `.onload()`)
- When demoting the Sub-App from Full Screen Mode to Preview Mode

Syntax

```
subApp.onPreview();
```

8.3. .onFullScreen()

The `.onFullScreen()` event is fired when the Run3TV (Main) Application moves the Sub-App into Full Screen Mode.

Syntax

```
subApp.onFullScreen();
```

8.4. .onNavNext() | .onNavBack() | .onNavUp() | .onNavDown() | .onNavEnter() | .onNavExit()

The following events are fired when the Sub-App is in Full Screen Mode and the viewer presses keys:

Function	Description
`.onNavNext()`	Fired when the viewer presses **RIGHT**
`.onNavBack()`	Fired when the viewer presses **LEFT**
`.onNavUp()`	Fired when the viewer presses **UP**
`.onNavDown()`	Fired when the viewer presses **DOWN**
`.onNavEnter()`	Fired when the viewer presses **OK** or **SELECT**
`.onNavExit()`	Fired when the viewer presses **BACK**

These events will not be fired in Preview Mode.

Syntax

```
subApp.onNavNext();
subApp.onNavBack();
subApp.onNavUp();
subApp.onNavDown();
subApp.onNavEnter();
subApp.onNavExit();
```

8.5. .exitFullScreen()

The `.exitFullScreen()` function is called by the Sub-App when it wishes to exit Full Screen mode.

This is likely to be called when the viewer indicates that they have finished interacting with the Sub-App, for example by pressing the **BACK** key.

Once this function has been called, the Run3TV (Main) Application will demote the Sub-App to Preview Mode.

Calling this function when in Preview Mode has no effect.

Syntax

```
subApp.exitFullScreen();
```

8.6. .isFullScreen

The `.isFullScreen` read-only parameter is queried by the Sub-App when it wishes to know the current mode that it is running in.

This parameter returns `true` if the Sub-App is currently in Full Screen mode, otherwise it returns `false`.

Syntax

```
boolean subApp.isFullScreen;
```

8.7. .getPreviewWindowScale()

The `.getPreviewWindowScale()` function is called by the Sub-App when it needs to know the size, position or scaling factor of the Preview window.

This function will provide the same values, no matter the current Mode of the Sub-App.

Syntax

```
subApp.getPreviewWindowScale(callbackFn(PreviewModeRenderingDetailsObject));
```

Parameters

`callbackFn`

A function to execute containing the response. The function is called with a single object containing the following fields:

> **`PreviewModeRenderingDetailsObject`**
>
> A response object containing the following fields:
>
> **`xPos : <Number (0.0 … 100.0)>`**
>
> The X-axis location of the Preview Mode window, as a percentage of full screen width
>
> **`yPos : <Number (0.0 … 100.0)>`**
>
> The Y-axis location of the Preview Mode window, as a percentage of full screen width
>
> **`scaleFactor : <Number(10.0 … 100.00)>`**
>
> The scaling factor of the Preview Mode window (as defined by the `org.atsc.scale-position` JSON-RPC WebSocket message in [A/344]).

Return value

None

Basic usage

The example below prints the response object of `getPreviewModeWindowScale()` to the console log.

```
subApp.getPreviewWindowScale(function(resultObject)
      {
        console.log(JSON.stringify(resultObject, null, "  "));
      }
    );


// {
//     “xPos” : 10.0,
//     “yPos” : 10.0,
//     “scaleFactor” : 25.0
// }
```

Annex A - Example Sub-Apps

This annex describes the example Sub-Apps available within the Starter Kit.

The example Sub-Apps are available in the `public/london/apps/` directory and can be explored on screen via the Q-bar's "Sub-Apps" content rail.

For access to the Starter Kit, please contact your Run3TV representative.

Note Some of the Sub-Apps provided in the Starter Kit are available in both English (`en.html`) and Spanish (`es.html`). Their functionality is equivalent.

run3tv-sub-app-basic

This Sub-App is an example of the near-minimum amount of code required to create a Sub-App.

It captures events from `subApp.run3tv.min.js`, as summarized below:

Action	Causes Event	Description
The Sub-App is launched	`N/A`	A weather map image is loaded
The Run3TV (Main) App informs the Sub-App that it has been launched	⇒ `.onLoad()`	Reports to the console
The Sub-App enters Preview Mode	⇒ `.onPreview()`	Reports to the console
The Sub-App enters Full Screen Mode	⇒ `.onFullScreen()`	Reports to the console
The **BACK** key is pressed	⇒ `.onNavExit()`	Reports to the console and calls `.exitFullScreen()` to inform the Run3TV (Main) Application that the viewer has completed navigation

run3tv-sub-app-video

This Sub-App is an example of a Sub-App with RSS-Configured Video associated with it.

It acts on the same events as detailed in run3tv-sub-app-basic.

Instead of presenting a weather image, it does not render across its entire window. This enables the video (presented by the AMP) to be visible. For more details about Z-ordering of the AMP and Sub-App see Sub-Apps With Video.

run3tv-sub-app-navigation

This Sub-App is an example of a Sub-App with RSS-Configured Video associated with it.

The table below summarizes the functionality of this Sub-App based on the events that it captures:

Action	**Causes Event**	**Description (Excluding console logging)**
The Run3TV (Main) App informs the Sub-App that it has been launched	⇒ `.onLoad()`	• Reports the preview screen scaling information to the console
The Sub-App enters Preview Mode	⇒ `.onPreview()`	• Updates the content of the text box on screen
The Sub-App enters Full Screen Mode	⇒ `.onFullScreen()`	• Updates the contents of text box on screen
The viewer presses **RIGHT**	⇒ `.onNavNext()`	• The text box is shifted 10 pixels in the direction of the arrow key chosen
The viewer presses **LEFT**	⇒ `.onNavBack()`	
The viewer presses **UP**	⇒ `.onNavUp()`	
The viewer presses **DOWN**	⇒ `.onNavDown()`	
The viewer presses **OK** / **SELECT**	⇒ `.onNavDown()`	• Changes the color of the text
The viewer presses **BACK**	⇒ `.onNavExit()`	• Calls `.exitFullScreen()`, informing the Run3TV (Main) Application that this Sub-App wishes to exit Full Screen Mode. • The Main Application will return the Sub-App to Preview Mode

run3tv-sub-app-video-control

This Sub-App is an example of a Sub-App with RSS-Configured Video associated with it. It provides play/pause video control functionality to the viewer.

The table below summarizes the functionality of this Sub-App based on the events that it captures:

Action	**Causes Event**	**Description (Excluding console logging)**
The Run3TV (Main) App informs the Sub-App that it has been launched	⇒ `.onLoad()`	• Reports the preview screen scaling information to the console
The Sub-App enters Preview Mode	⇒ `.onPreview()`	• Updates the content of the text box on screen • Sets the visibility of the Play/Pause button to hidden, so that it is removed from the screen when the Sub-App moves from Full Screen Mode back to Preview Mode
The Sub-App enters Full Screen Mode	⇒ `.onFullScreen()`	• Updates the contents of text box on screen • Presents the Play/Pause button
The viewer presses **RIGHT**	⇒ `.onNavNext()`	• The text box is shifted 10 pixels in the direction of the arrow key chosen
The viewer presses **LEFT**	⇒ `.onNavBack()`	
The viewer presses **UP**	⇒ `.onNavUp()`	
The viewer presses **DOWN**	⇒ `.onNavDown()`	
The viewer presses **OK / SELECT**	⇒ `.onNavDown()`	• Toggles play/pause by calling either `.ampPlay()` or `.ampPause()` depending on the playback state Note The Play/Pause button text is not updated here. Instead it is updated based on the `.ampPlaybackStateChange()` event.
The viewer presses **BACK**	⇒ `.onNavExit()`	• `.exitFullScreen()` is called, informing the Run3TV (Main) App that this Sub-App wishes to exit Full Screen Mode. • The Main Application will return the Sub-App to Preview Mode
The state of AMP playback has changed	⇒ `.ampPlayback StateChange()`	• Updates the Play/Pause button text