



NextGen TV Run3TV Implementation Guidelines: Run3TV Application Management

Version: 1.0 (Released)
Date: 19 October 2023
Doc: PNC-IG-0204

© 2023 Pearl TV, LLC and A3FA, LLC.

Confidential. All rights reserved.

For the Pearl Network Consortium.

Revision history

Version	Date	Update
1.0 (Released)	19 Oct 2023	Initial draft for Framework v2.0 release

Copyright notice

This document is copyright © 2021 - 2023 Pearl TV, LLC and should not be revised, modified, redistributed or republished in whole or in part, without the express written permission of Pearl TV, LLC.

The “RUN3TV” name and logos are registered servicemarks of A3FA, LLC, with all rights reserved.

The rights of the creators of all specifications and trademarks and servicemarks referenced within this document are fully acknowledged and must be respected in the application of this specification.

Contents

Revision history	2
Copyright notice	2
Contents	3
1. Glossary	5
2. References	5
3. Introduction	6
4. Framework configuration overview	8
4.1. File references within the Framework configuration files	9
4.2. Business rules	9
4.2.1. Structure, presence and format of Framework configuration files	9
5. Linking channels to Broadcast Applications (bridge.json)	10
5.1. Business rules	12
5.1.1. Structure, presence and format of bridge.json	12
5.1.2. Updates to bridge.json	12
5.1.3. Top level bridge.json fields	12
5.1.4. <GSID> entries in bridge.json	13
5.1.5. Service Object entries in bridge.json	13
5.1.6. ServiceInfo level bridge.json fields	15
5.2. bridge.json structure	15
5.2.1. Root	15
5.2.2. Service Object	16
5.3. Files referenced by bridge.json	17
6. Definition of applications (appsList.json)	18
6.1. Business rules	19
6.1.1. Structure of appsList.json	19
6.1.2. Updates to appsList.json	19
6.1.3. Top level and secondary level appsList.json fields	20
6.1.4. Legal Object entries in bridge.json	21
6.1.5. Application Object entries in bridge.json	22
6.2. appsList.json structure	24
6.2.1. Root	24
6.2.2. Legals Object	25
6.2.3. Terms of service and privacy policy files	25
6.2.4. Applications Object	26
6.2.5. Application Properties	28

7. Configuring applications (config.json)	30
7.1. Substitution with Replaceable Field Tags	32
7.2. config.json structure	34
7.2.1. Root	34
7.2.2. Multi language string object	35
8. Application scheduling for a channel (appSchedule.json)	36
8.1. Scheduling flow	37
8.2. Scheduling scenarios	40
8.3. Precision	40
8.4. Invalid appSchedule.json files	41
8.5. Business rules	41
8.5.1. Structure of appsList.json	41
8.5.2. Updates to appSchedule.json	42
8.5.3. Top level and secondary level appSchedule.json fields	43
8.5.4. Schedule Object entries in appSchedule.json	43
8.6. appSchedule.json structure	45
8.6.1. Root	45
8.6.2. Schedule Object	45
ANNEX A - bridge.json example	46
ANNEX B - appsList.json example	48
ANNEX C - appSchedule.json example	49

1. Glossary

Glossary of unfamiliar words and acronyms.

Term	Definition
AMP	Application Media Player (ATSC 3.0)
BA	Broadcast Application (ATSC 3.0)
BCP 47	A set of standardized codes used to identify human languages. The Framework allows for either EN (English) or ES (Spanish) language codes to be used.
DMA	Designated Market Area (DMA®). DMA® is a registered service mark of The Nielsen Company.
RMP	Receiver Media Player (ATSC 3.0)

2. References

ID	Publisher	Document
A344-2023-02	ATSC	ATSC 3.0 Interactive Content, A/344:2023-02, 17 February 2023
DASH-IF-IOP-5	DASH	DASH-IF IOP-5, v5.0.0 (2021-11)
PNC-IG-0231	A3FA	RSS Feed Specification
PNC-IG-0232	A3FA	AEAT and OSN Feed Specification
PNC-IG-0233	A3FA	ESG (OMA) Feed Specification

3. Introduction

The RUN3TV A3FA Framework is a suite of software modules written to support the creation and deployment of ATSC 3.0 Broadcaster Applications (BAs), offering:

- Application Lifecycle Management
- Common components and controls such as Application Media Player (AMP) and control of the Receiver Media Player (RMP)
- Predefined menu structures and widgets
- Common APIs for event notifications

This document describes the application lifecycle management features of the RUN3TV A3FA Framework v2.

The Framework enables developers of ATSC 3.0 BAs to configure the features and presentation of the application(s). It also enables developers to configure application lifecycles in the following ways:

- Securely **associate** one or more BAs to individual or multiple ATSC 3.0 channels
- Securely **schedule** one or more BAs on individual or multiple ATSC 3.0 channels

It is designed to support simple, swift setup by offering configuration options at both the market and the station level.

This document describes the product features and the various file formats/interface standards for application lifecycle management. As such, it is intended to be read by ATSC 3.0 product managers and software developers.

It is expected that the reader will have a firm understanding of ATSC 3.0 BA development and access to the RUN3TV A3FA Starter Kit.

If you do not have access to the Starter Kit, please contact your A3FA representative.

The RUN3TV A3FA Framework offers a set of applications ready to configure and deploy.

The starting point for associating applications with services is the **bridge.json** file, which is described in detail in the following sections.

Throughout this document, business rules sections provide detailed information on the functional and non-functional requirements of the Framework.

A “traffic light” Impact coding is used to indicate successful and unsuccessful paths, as summarized below:

Impact	Description
<i>Green</i>	The Framework is able to provide functionality to all services
<i>Amber</i>	Due to a fault, the Framework is unable to provide functionality to one or more services, or functionality on these services is limited or erroneous
<i>Red</i>	The Framework is unable to provide functionality to any service

4. Framework configuration overview

The A3FA Framework uses a small number of .json files to allow flexible configuration of one or more applications and services:

bridge.json associates applications (with schedules and configurations) to services. It references the following configuration files

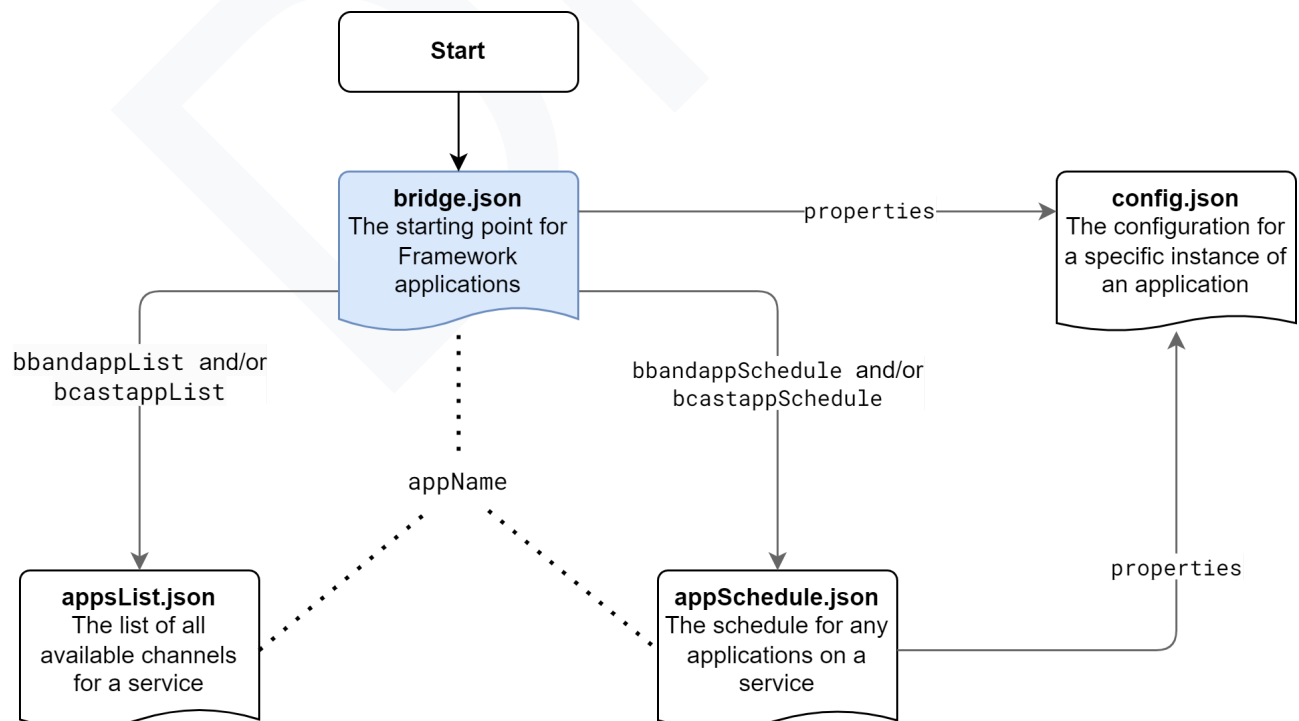
- **appsList.json** - a list of all possible applications
- **appSchedule.json** - a services' schedule of applications
- **config.json** - an application's configuration information

Within these .json files, applications are cross-referenced using the appName field.

All text files within the A3FA Framework, including these .json files are encoded using UTF-8.

Note AppName Ids are case sensitive. If a mistyped appName is used (for example, in **appsList.json** an application exists named “*BigEvent*” however in **appSchedule.json**, it is referred to as “*Bigevent*”, no application will launch. Instead, an error will be presented in the web browser console.

The diagram below summarizes how the various .json files link together.



4.1. File references within the Framework configuration files

Throughout the configuration files, references to other files can be either relative, absolute or HTTPS links, as summarized below.

Type	Description	Example
Relative	A unix file location within the filesystem of the framework, based on the location of the file that references it	../london/qbar/config.json
Absolute	A unix location within the filesystem of the framework, based from the root of the framework filesystem	/london/qbar/config.json
HTTPS	A file hosted on an HTTPS server, accessible by receivers	https://example.com/config.json

4.2. Business rules

4.2.1. Structure, presence and format of Framework configuration files

Req	Scenario	Impact	Description
framework-St-1	The configuration file is present, valid, and well formed	None	The Framework shall parse this file and configure the service(s), app(s) or schedule(s) as defined
framework-St-2	The configuration file is present, valid and well formed, but contains additional data fields	None	The Framework shall parse this file and configure the service(s), app(s) or schedule(s) as defined. It shall ignore any fields that it does not expect
framework-St-3	The configuration file is present, valid and well formed, but contains invalid UTF-8 data	Various	<p>The Framework shall parse this file and configure the service(s), app(s) or schedule(s) as defined where possible.</p> <p>If the invalid data sits within a primary key field (such as the GSID field in bridge.json), or within a cross-reference field (such as the appName field), the Framework will either mis-configure itself based on the incorrect data or be unable to configure itself for that particular feature.</p> <p>The Framework shall present an error in the console log where possible and launch the exception application defined in the fmw.conf.</p> <p>Note For more details fmw.conf, see <TODO>.</p>

5. Linking channels to Broadcast Applications (bridge.json)

Associating applications to ATSC 3.0 channels is performed using the **bridge.json** file supplied in the Starter Kit at:

```
/public/bridge-conf/1.0/bridge.json
```

This configuration file lists each ATSC 3.0 Global Service ID (GSID) and associates that GSID with a list of applications and an optional associated schedule. The Framework uses this schedule to select the correct application to launch at the right time. If no schedule is provided, or all schedule events are set in the future, the Framework allows for a configurable default application to be associated with the GSID.

Broadcasters must ensure that their ATSC 3.0 service information configuration is set up to correctly launch the Framework for each GSID that they manage. Refer to [PNC-IG-0202] for more details.

Note If the broadcaster wishes to schedule applications, the **appSchedule.json** file can be used. See [Application scheduling for a channel \(appSchedule.json\)](#)

If the broadcaster only requires a single application to be available at all times, no schedule file is required. In this case, the `defaultApp` field determines the application to be associated with the service.

A simplified example **bridge.json** file is shown below.

```

1 {
2   "connection": "bband",
3   "bridge": {
4     "https://doi.org/10.5238/8111-1111": {
5       "bbandappList": "/london/appsList.json",
6       "appDefault": {
7         "appName": "field-test",
8         "properties": {}
9       },
10      "serviceInfo": {
11        "broadcaster": "Yotta",
12        "network": "Run3TV",
13        "stationDMACode": 999,
14        "callSign": "RYML-2"
15      }
16    },
17    "https://doi.org/10.5239/8222-2222": {
18      "bcastappList": "/london/appsList.json",
19      "bcastappSchedule": "/london/Station-1/schedule/appSchedule.json",
20      "appDefault": {
21        "appName": "cta-ba",
22        "properties": {}
23      },
24      "serviceInfo": {
25        "broadcaster": "Yotta",
26        "network": "Run3TV",
27        "stationDMACode": 999,
28        "callSign": "RYML-3"
29      }
30    }
31  }
32 }

```

Note A fully populated example bridge.json file can be found in Annex A.

This example **bridge.json** file describes two services:

The service with a GSID of "https://doi.org/10.5238/8111-1111":

- Contains a list of applications available for launch via broadband (line 5).
- Has a default application named "field-test" (line 7)
- Ensures reporting and analytics are correctly attributed (lines 11 - 14)

The service with GSID of "https://doi.org/10.5239/8222-2222":

- Contains a list of applications available for launch via broadcast (line 18)
- Has a schedule for when these applications should run (line 19)
- Has a default application named "cta-ba" (line 21)
- Ensures reporting and analytics are correctly attributed (lines 25 - 28)

The detailed structure of **bridge.json** is described in the following section.

5.1. Business rules for bridge.json

5.1.1. Structure, presence and format of bridge.json

Req	Scenario	Impact	Description
bridge-St-1	The bridge.json file is either missing, mis-named or mislocated in the Framework's directory structure	Exception app loaded on all services	The Framework shall present an error in the console log. The exception application will be launched. This application contains no viewer-facing assets. For more details, see PNC-IG-0207
bridge-St-2	The bridge.json file is present, but does not contain valid, well formed json data	Exception app loaded on all services	The Framework shall present an error in the console log. The exception application will be launched. This application contains no viewer-facing assets. For more details, see PNC-IG-0207

5.1.2. Updates to bridge.json

Req	Scenario	Impact	Description
bridge-Up-1	The bridge.json file is updated	Various	The Framework shall load the bridge.json file once, when first launching. In general, the framework will be launched when the viewer: <ul style="list-style-type: none"> Tunes to a Framework-enabled service: <ul style="list-style-type: none"> From a service on a different multiplex From a service on the same multiplex that is not configured to persist the Framework Exits the EPG (or similar section of terminal UI) back to the Framework-enabled service

5.1.3. Top level bridge.json fields

Req	Scenario	Impact	Description
bridge-TL-1	The connection field is present and contains a valid enum (currently only "bband" is a permitted value)	—	The Framework shall parse this file and configure the service(s) as defined
bridge-TL-2	The connection field is present and contains an invalid enum	—	The Framework shall parse this file and configure the service(s) as defined. The Framework will treat the absence of a valid connection field as the equivalent to: connection = bband
bridge-TL-3	The connection field is missing	—	The Framework shall parse this file and configure the service(s) as defined. The Framework will treat the absence of a valid connection field as the equivalent to: connection = bband
bridge-TL-4	The bridge field is present and contains a valid data structure	—	The Framework shall parse this file and configure the service(s) as defined.
bridge-TL-5	The bridge field is present and contains an invalid data structure	Exception app loaded on all services	The Framework shall stop parsing the file and present an error in the console log. The exception application will be launched. This application contains no viewer-facing assets. For more details, see PNC-IG-0207

Req	Scenario	Impact	Description
bridge-TL-6	The bridge field is missing	Exception app loaded on all services	The Framework shall stop parsing the file and present an error in the console log. The exception application will be launched. This application contains no viewer-facing assets. For more details, see PNC-IG-0207

5.1.4. <GSID> entries in bridge.json

This section describes the business rules for Global Service ID entries in the bridge.json file.

Req	Scenario	Impact	Description
bridge-GS-1	The <GSID> field is present and contains a valid GSID that is in the receiver's service list	—	On detecting that this GSID is present in the receiver's channel list, the Framework shall associate any valid data within this <GSID> object with the service that has the same GSID value
bridge-GS-2	The <GSID> field is present and contains a valid GSID that is not in the receiver's service list	—	On detecting that this GSID is not in the receiver's channel list, the Framework shall ignore any configuration within this <GSID> object
bridge-GS-3	The <GSID> field is present and contains an invalid GSID value	Exception app loaded on the service	On detecting that this (invalid) GSID is not in the receiver's channel list, the Framework shall ignore any configuration within this <GSID> object. The Framework will continue to parse the bridge.json file to associate any other valid and known <GSID> fields with applications. If the viewer tunes to the service whose GSID was incorrectly entered into bridge.json, the Framework will launch the exception application. This application contains no viewer-facing assets. For more details, see PNC-IG-0207. The Framework will report this to the console log
bridge-GS-4	The <GSID> field is present and is a duplicate of another <GSID> value	Exception app loaded on all services	A file containing duplicate <GSID> values is invalid JSON. The Framework shall stop parsing the file and present an error in the console log. The exception application will be launched. This application contains no viewer-facing assets. For more details, see PNC-IG-0207
bridge-GS-5	The <GSID> field is present and valid, but does not contain valid data within	Exception app loaded on the service	On detecting this invalid data, the Framework shall ignore any configuration within this <GSID> object. The Framework will continue to parse the bridge.json file to associate any other valid and known <GSID> fields with applications If the viewer tunes to the service whose GSID object was incorrectly populated, the Framework will launch the exception application. This application contains no viewer-facing assets. For more details, see PNC-IG-0207. The Framework will report this to the console log
bridge-GS-6	The broadcaster has not performed the correct ATSC 3.0 headend configuration to associate the Framework with a service	No application on intended service	The receiver will not attempt to launch the Framework on this service
bridge-GS-7	The broadcaster has correctly configured the ATSC 3.0 headend to associate the Framework with the service, however the bridge.json file does not include a matching <GSID>	Exception app loaded on the service	The Framework shall launch on this service, however will not locate any configuration information for the service. The Framework will report this to the console log. The exception application will be launched on this service. This application contains no viewer-facing assets. For more details, see PNC-IG-0207
bridge-GS-8	The receiver has no internet connection (or that internet connection does not have a route to access the Framework)	No application on any service	The receiver will be unable to download the Framework

Req	Scenario	Impact	Description
bridge-GS-9	The broadcaster has not signaled the Framework on any of the ATSC 3.0 PLP(s) that the receiver can access	No application on intended service	The receiver will not attempt to launch the Framework on the this service

5.1.5. Service Object entries in bridge.json

This section describes the business rules for Global Service ID entries in the bridge.json file.

Req	Scenario	Impact	Description
bridge-SO-1	The bbandappsList field is present and contains a valid file reference to a local Framework file or https location accessible to the receiver	—	The Framework will locate the referenced appslist.json file and parse it to collect all applications available for the service
bridge-SO-2	The bbandappsList field is either not present, or contains invalid data (for example, an integer instead of a file reference)	Exception application on intended service	The Framework will skip the current Service Object. No application will be associated with the GSID of that object. The Framework will report this to the console log. The exception application will be launched. This application contains no viewer-facing assets. For more details, see PNC-IG-0207
bridge-SO-3	The bbandappsList field refers to a file on an HTTPS website that is not available to the receiver. This includes files that are only available to logged in users of that website	Exception application on intended service	The Framework will skip the current Service Object. No application will be associated with the GSID of that object The Framework will report this to the console log The exception application will be launched. This application contains no viewer-facing assets. For more details, see PNC-IG-0207
bridge-SO-4	The appDefault field is present and it contains a valid, well formed object (of appName and properties fields)	—	The Framework will search for the application named in the appDefault / appName field in the appsList.json file, and if found, launch it on the service. If the properties field is an empty object, the application will take its properties information from the appsList.json file Otherwise, the application will take its properties information solely from the information in the appDefault / properties field. In other words, it is not possible to selectively override the application's configuration by only supplying a subset of configuration values in the properties field If a separate schedule is provided for this service, and that schedule contains an event with a start timestamp in the past, that scheduled event will determine the application and configuration to launch, not this defaultApp object. For more information, see Business rules for appSchedule.json
bridge-SO-5	The appDefault field exists but does not contain a well formed object	Various	The Framework will report an error to the console log. If a separate schedule is provided for this service, and that schedule contains an event with a start timestamp in the past, that scheduled event will determine the application and configuration to launch on the service. Otherwise, the Framework shall present an error in the console log and the exception application will be launched. This application contains no viewer-facing assets. For more details, see PNC-IG-0207

Req	Scenario	Impact	Description
bridge-SO-6	The appDefault field is present and it contains a valid, well formed object, however the child appName refers to an application that does not exist in the appsList.json file	Variable	<p>The Framework will search for the application named in the child appName field in the appsList.json file. On failing to find the application, it will report an error to the console log.</p> <p>Note appName fields are case sensitive. For example, "TestApp" and "TESTAPP" do not refer to the same application.</p> <p>If a separate schedule is provided for this service, and that schedule contains an event with a start timestamp in the past, that scheduled event will determine the application and configuration to launch on the service. Otherwise, the Framework shall present an error in the console log and the exception application will be launched. This application contains no viewer-facing assets. For more details, see PNC-IG-0207</p>
bridge-SO-7	The appDefault field is present and it contains a valid, well formed object with a known application in the child appName field. However the properties field is not well formed	Variable	<p>The Framework does not perform validation of the properties field. Instead it passes the properties field to the application for validation. It is the application's responsibility to either fail gracefully or degrade functionality gracefully</p>
bridge-SO-8	The appDefault field is present and it contains a valid, well formed object with a known application in the child appName field. However the properties field contains a reference to a config.json file that is not reachable by the receiver	Variable	<p>If a separate schedule is provided for this service, and that schedule contains an event with a start timestamp in the past, that scheduled event will determine the application and configuration to launch on the service. Otherwise, the Framework will attempt to download the config.json file from the properties field. On discovering that this config.json is not available, the Framework shall present an error in the console log and the exception application will be launched. This application contains no viewer-facing assets. For more details, see PNC-IG-0207</p>

5.1.6. ServiceInfo level bridge.json fields

Req	Scenario	Impact	Description
bridge-SL-1	The serviceInfo field is present and contains a valid, well formed object with data that correctly defines the GSID's service	–	<p>The Framework will follow its standard process of selecting an application from either the bridge.json defaultApp field or the appSchedule.json file (if present).</p> <p>Applications associated with this service will make use of the information provided in the serviceInfo field to populate analytics and reporting responses</p>
bridge-SL-2	The serviceInfo field is not present, or its content is not valid and well formed	No application on intended service	<p>The Framework will skip the current Service Object. No application will be associated with the GSID of that object</p> <p>If the viewer tunes to this service, the Framework shall present an error in the console log and the exception application will be launched. This application contains no viewer-facing assets. For more details, see PNC-IG-0207</p>
bridge-SL-3	The serviceInfo field contains incorrect information	Incorrect reporting for service	<p>The Framework will accept the incorrect values and provided then when required as part of analytics and reporting responses</p>

5.2. bridge.json structure

This section describes the bridge.json data structure

5.2.1. Root

The following fields are required, unless otherwise stated.

Field	Description	Required / Type	Example
connection	An enum to distinguish between applications delivered via the ATSC 3.0 NRT channel and those delivered via broadband. The current version of the Framework permits a single value: <ul style="list-style-type: none"> bband - broadband Future versions of the framework shall expand upon this list	Optional String (default: bband)	bband
bridge	A listing of GSIDs and their associated Broadcast Application(s)	Object	{...}
<GSID>	One or more fields, each of which containing a valid, complete and unique GSID value	Service Object	https://doi.org/10.5239/8222-2222

5.2.2. Service Object

The following fields are required, unless otherwise stated.

Field	Description	Required / Type	Example
bbandappList	A file location within the Starter Pack to the appsList.json file. This file shall contain all possible applications available via broadband for the service. <i>bbandappList</i> must list only broadband-provided BAs	At least one required String (path)	/london/appsList.json or https://example.com
bcstappList	Reserved for future use		
bbandappSchedule	A file location either within the Starter Pack or accessible via the internet to an appSchedule.json file that provides a schedule for the applications listed by <i>bbandappList</i> . <i>bbandappSchedule</i> shall only be present if the <i>bbandappList</i> field exists	Optional String (path)	/london/appSchedule.json or https://example.com/appSchedule.json
bcstappSchedule	A file location either within the Starter Pack or accessible via the internet to an appSchedule.json file that provides a schedule for the applications listed by <i>bcstappList</i> . <i>bcstappSchedule</i> shall only be present if the <i>bcstappList</i> field exists	Optional String (path)	/london/appSchedule.json or http://example.com/appSchedule.json
appDefault	The default application to launch. If no schedule is provided, this application shall be associated with the service	Object	{ "appName": "cta-ba", "properties": "config.json" }

Field	Description	Required / Type	Example
appName	The name of the default application. This name must exist within the appsList.json file linked in the <i>bbandappList</i> field. Application names cannot contain spaces.	String (App Name)	cta-ba
properties	Properties for the default application. The data here will fully replace any configuration in the application properties defined in the appSchedule.json file. Implementation of the properties field has changed in v2.0 of the Framework. For more details, see the Application Properties section.	Object / Local File Reference	"config.json"
serviceInfo	The reporting and analytics configuration for the service	Object	
broadcaster	The name of the broadcaster	String	Hearst
network	The name of the broadcast network	String	NBC
stationDMACode	The Designated Market Area (DMA) code of the broadcaster	Integer	223
callsign	The callsign of the broadcaster	String	KGET-NG

5.3. Files referenced by bridge.json

The bridge.json file must reference at least one **appList.json** file within either the *bbandappList* and/or *bcastappList* fields.

If application schedules are required, **appSchedule.json** files must be referenced in either the *bbandappSchedule* and/or *bcastappSchedule* fields.

These file references can be either absolute, relative or a URL.

Both file types are described in detail below.

6. Definition of applications (appsList.json)

A list of all available applications, their versions, and other metadata must be included in an **appsList.json** file.

A simplified example **appsList.json** file is shown below.

```

1  {
2    "appList": {
3      "legals": {
4        "default": {
5          "match-keys": {
6            "name": "COMPANY 12",
7            "address": "Company Address No. 12",
8            "email": "contact@domain.net",
9            "date": "Tuesday, December 08, 2020"
10         },
11         "terms-of-service": "/run3tv-common/tos/terms-of-service.html",
12         "privacy-policy": "/run3tv-common/tos/privacy-policy.html"
13       },
14       "custom": {
15         "match-keys": {
16           "company": "COMPANY 21",
17           "address": "Company Address No. 21"
18         },
19         "terms-of-service": "/london/Station-2/common-app/tos/terms-of-service.html",
20         "privacy-policy": "/london/Station-2/common-app/tos/privacy-policy.html"
21       }
22     },
23     "applications": {
24       "field-test": {
25         "appRoot": "/run3tv-common/apps",
26         "appEntry": "/field-test.html",
27         "appVersion": "1.0.0",
28         "appId": "com.run3tv.fieldtest",
29         "properties": "/london/field-test/q-bar/dynamic/menu/config.json"
30       },
31       "STATION-1": {
32         "appRoot": "/run3tv-common/apps/common-app",
33         "appEntry": "/index.html",
34         "appVersion": "1.0.0",
35         "appId": "com.run3tv-common/q-bar",
36         "properties": "/london/Station-1/q-bar/dynamic/menu/config.json",
37         "allowInterAppStorage": true,
38       }
39     }
40   }
41 }
42 }
43 }
```

This example **appsList.json** file describes two applications available (to be associated with GSIDs in the **bridge.json** file).

The [Applications Object](#) (lines 23 - 41) describes all possible applications that could be associated with the GSIDs in the **bridge.json** file. The example above describes two applications:

- field-test
- STATION-1

Both applications include the required configuration fields and properties needed to successfully locate, launch and configure the application.

The two [Legals Objects](#) (lines 3 - 22) contain legal information that is inserted into each application, such as company name, terms of service and privacy policy. This information is available to the viewer in the Settings section of the application (where present).

Within each Legal Object, the `match-keys` set of key-values can be used to dynamically insert additional data into the terms-of-service and privacy-policy text. Further details are provided below.

The data within the “default” Legals Object is used if an application does not contain a reference to a specific “legals” field within its configuration using the `legal` field.

For example, within the properties of the STATION-1 application (line 36, above), a line such as shown below would associate the “custom” legal object to that application.

STATION-1 example configuration	
config.json	<pre>{ ... "legal": "custom", ... }</pre>

The following sections describe the structure of **appsList.json** in detail.

6.1. Business rules for appsList.json

6.1.1. Structure of appsList.json

Req	Scenario	Impact	Description
AppsList-St-1	The appsList.json file is present, valid, and well formed	—	The Framework shall parse this file and configure the service(s) as defined
AppsList-St-2	The appsList.json file is either missing, mis-named or mislocated	Exception app loaded on all services	The Framework shall stop parsing the file and present an error in the console log. The exception application will be launched. This application contains no viewer-facing assets. For more details, see PNC-IG-0207
AppsList-St-3	The appsList.json file is present, but does not contain valid, well formed json data	Exception app loaded on all services	The Framework shall stop parsing the file and present an error in the console log. The exception application will be launched. This application contains no viewer-facing assets. For more details, see PNC-IG-0207
AppsList-St-4	The appsList.json file is present, valid and well formed, but contains additional data fields	—	The Framework shall parse this file and configure the service(s) and applications as defined. It shall ignore any fields that it does not expect

6.1.2. Updates to appsList.json

Req	Scenario	Impact	Description
AppsList-Up-1	The appsList.json file is updated to remove the application that the Framework is currently presenting to the viewer	Various	<p>The Framework shall load the appsList.json file once, when first launching. In general, the framework will be launched when the viewer:</p> <ul style="list-style-type: none"> Tunes to a Framework-enabled service: <ul style="list-style-type: none"> From a service on a different multiplex From a service on the same multiplex that is not configured to persist the Framework Exits the EPG (or similar section of terminal UI) back to the Framework-enabled service

6.1.3. Top level and secondary level appsList.json fields

Req	Scenario	Impact	Description
appsList-TL-1	The appsList field is present and contains valid, well formed data	—	The Framework shall parse this file and configure itself for the application(s) as defined
appsList-TL-2	The legals field is present and contains invalid or non-well formed data	No application on intended service	The Framework shall stop parsing the file and present an error in the console log. No application will be associated with any service that makes use of this file
appsList-TL-3	The legals field is missing	No application on intended service	The Framework shall stop parsing the file and present an error in the console log. No application will be associated with any service that makes use of this file
appsList-TL-4	The applications field is present and contains valid, well formed data, including at least one Application Object	—	The Framework shall parse this file and configure itself for the application(s) as defined.
appsList-TL-5	The applications field is present and contains an invalid data structure	No application on intended service	The Framework shall stop parsing the file and present an error in the console log. The exception application will be launched. This application contains no viewer-facing assets. For more details, see PNC-IG-0207
appsList-TL-6	The applications field is missing, or no Application Objects exist within this field	No application on intended service	The Framework shall stop parsing the file and present an error in the console log. The exception application will be launched. This application contains no viewer-facing assets. For more details, see PNC-IG-0207

6.1.4. Legal Object entries in bridge.json

This section describes the business rules for Legals Object entries in the bridge.json file.

Req	Scenario	Impact	Description
appsList-LO-1	The legals/default field is present and contains valid, well formed data	—	<p>The Framework shall parse this file and configure itself for the application(s) as defined.</p> <p>Any application that does not include an <AppName> / legals field, or includes this field with the value "default" will be associated with this legal information</p>

Req	Scenario	Impact	Description
appsList-LO-2	The legalIDs field is present and contains invalid or non-well formed data	Variable	The Framework shall parse this file and configure itself for any applications that do not rely on the default legal data. Any applications that rely on the default legal data will not be configured, and will not be made available for association with a service. The Framework shall present an error in the console log
appsList-LO-3	The legalIDs/default field is missing	Variable	The Framework shall parse this file and configure itself for any applications that do not rely on the default legal data. Any applications that rely on the default legal data will not be configured, and will not be made available for association with a service. The Framework shall present an error in the console log
appsList-LO-4	An application references (using the <appName>/legalIDs field) a legalIDs/<legalID> field that exists and contains valid, well formed data	–	The Framework shall associate the the legal information provided in the legalIDs/<legalID> object
appsList-LO-5	An application references (using the <appName>/legalIDs field) a legalIDs/<legalID> field that does not exist	Exception app loaded on service	The Framework shall stop parsing this application's configuration, and shall not make this application available to be associated with a service. The Framework shall present an error in the console log. The exception application will be launched. This application contains no viewer-facing assets. For more details, see PNC-IG-0207
appsList-LO-6	An <appName>/legalIDs field exists that shares the same <legalID> as another <legalID> field	Exception app loaded on all services	A file containing duplicate <legalID> values is invalid JSON. The Framework shall stop parsing the file and present an error in the console log. The exception application will be launched. This application contains no viewer-facing assets. For more details, see PNC-IG-0207
appsList-LO-7	A Legal Object contains zero, one or more of the following well formed fields: match-keys/name match-keys/address match-keys/email match-keys/date	–	The Framework shall store these keys and values, and shall use them as replacement tokens (referred to as match-keys) to populate data within the files referenced by the terms-of-service and privacy-policy fields in the current Legal Object. For more details, see Terms of service and privacy policy files . Each match-key can be used any number of times in either file. The use of each match-key is case sensitive. In other words, only the following match-key will be replaced within the terms of service and privacy policy files: {name} {address} {email} {date} These match-keys are only usable within the terms of service and privacy policy files. Use anywhere else in the Framework will result in the literal string of the match-key being retained
appsList-LO-8	A Legal Object contains a terms-of-service and a privacy-policy field. These fields both refer to either a file local to the Framework, or available to the receiver over the internet via HTTPS	–	The Framework accesses each file when requested to present their content to the viewer via the appropriate section of the application (where applicable). The HTTP context of where these files are inserted is within <body>. Prior to presenting the content, the Framework searches and replaces all instances of Match Keys (see appsList-LO-7) and Field Tags (see appsList-LO-B) with their substitution data
appsList-LO-9	A Legal Object contains a terms-of-service and a privacy-policy field. One or both of these files are on an HTTPS web server that is not reachable by the receiver	Application section not available	The Framework attempts to access the file when requested to present the content to the viewer. On failing to access this file, the Framework will present an error in the console log. The viewer will not be able to access the legal information in the application, and will be presented with a "This section is not available" error message

Req	Scenario	Impact	Description
appsList-LO-A	A Legal Object contains a terms-of-service and a privacy-policy field. One or both of these files contains invalid HTML, or HTML that does not match the context of the DOM into which it is inserted	Application section poorly presented	The Framework attempts to access the file when requested to present the content to the viewer. Other than attempting to replace any match-keys or Field Tags, the Framework performs no validation of the content provided. The receiver's browser will attempt to present the invalid content as well as possible. No error is sent to the console log
appsList-LO-B	A Legal Object contains a terms-of-service and a privacy-policy field. These fields both refer to accessible files that contain Replaceable Field Tags such as <callsign>	—	The Framework accesses each file when requested to present their content to the viewer via the appropriate section of the application (where applicable). Prior to presenting the content, the Framework searches and replaces all instances of Field Tags (see Substitution with Replaceable Field Tags) with their substitution data

6.1.5. Application Object entries in bridge.json

Req	Scenario	Impact	Description
appsList-AN-1	An Application Object contains a valid and unique <AppName> field and associated, valid, well formed and correctly configured object	—	The Framework parses the appsList.json file, detects this new application and makes it available to be associated with a service, either via the bridge.json file's defaultApp field, or via a schedule as defined by an appSchedule.json file. The Framework does not perform any validation of the usability of the application until it attempts to launch that application
appsList-AN-2	An Application Object contains a valid <AppName> field and associated, valid, well formed and correctly configured object, however the AppName ID field is a duplicate of another application	Application will not be associated with a service	The Framework parses the appsList.json file, detects that this appName ID is a duplicate. The Framework marks this appName ID as invalid and will present an error in the console log. This appName ID will be associated with a service, even if it is set the bridge.json file's defaultApp field, or via a schedule as defined by an appSchedule.json file
appsList-AN-3	An Application Object contains an invalid and unique <AppName> field and associated, valid, well formed and correctly configured object	Application will not be associated with a service	The Framework parses the appsList.json file, detects that this appName ID is invalid. The Framework discards this appName ID as invalid and will present an error in the console log. This appName ID will be associated with a service, even if it is set the bridge.json file's defaultApp field, or via a schedule as defined by an appSchedule.json file
appsList-AN-4	An Application Object contains valid appRoot and appEntry fields that can be resolved by the Framework to a usable application	—	The Framework parses these fields and stores them as part of the Application's definition. The Framework does not perform any verification that the appRoot and appEntry fields resolve to a usable application until it attempts to launch that application. The Framework it uses these fields to locate the application when required
appsList-AN-5	An Application Object contains valid appVersion field	—	The Framework parses these fields and stores them as part of the Application's definition. When the application is launched, the Framework uses this field as part of reporting and analytics The version may also be presented in the Settings section of the application, if available

Req	Scenario	Impact	Description
appsList-AN-6	An Application Object does not contain a valid, well formed appVersion field	Warning raised	The Framework will detect the absence of a valid appVersion field and present a warning in the console log. Any use of the appVersion value in the Framework will be substituted with an empty string
appsList-AN-7	An Application Object contains a valid appId field	—	The Framework will parse this field and store it as part of the Application's definition. When the application is launched, the Framework uses this field as part of reporting and analytics.
appsList-AN-8	An Application Object does not contain a valid, well formed appId field	Warning raised	The Framework will detect the absence of a valid appId field and present a warning in the console log. Any use of the appId value in the Framework will be substituted with an empty string
appsList-AN-9	An Application Object contains a valid properties field	—	The Framework permits either: <ul style="list-style-type: none"> an object of key/value entries describing the application's configuration, or, a file reference to a configuration file containing the key/value entries describing the application's configuration The Framework will parse this field and store it as part of the Application's definition. This configuration may be overridden if any of the following are true: <ul style="list-style-type: none"> The Framework puts this application live based on an appSchedule.json file (if present for the service) event with the same application name and a non-empty properties field The Framework puts this application live based on the bridge.json file defaultApp the same application name and a non-empty properties field
appsList-AN-A	An Application Object does not contain a well formed properties field (and the application's properties are not overridden by other files such as bridge.json or appSchedule.json)	Variable	The Framework does not perform validation of the properties field. Instead it passes the properties field to the application for validation. It is the application's responsibility to either fail gracefully or degrade functionality gracefully
appsList-AN-B	An Application Object contains a valid properties field referencing an HTTPS file that cannot be accessed by the receiver (and the application's properties are not overridden by other files such as bridge.json or appSchedule.json)	Exception app loaded on service	The Framework shall stop parsing this application's configuration, and shall not make this application available to be associated with a service. The Framework shall present an error in the console log. The exception application will be launched. This application contains no viewer-facing assets. For more details, see PNC-IG-0207

Req	Scenario	Impact	Description
appsList-AN-C	An Application Object contains a valid <i>allowInterAppStorage</i> field	–	<p>The Framework will parse this field and either allow or deny access to the following methods:</p> <pre>setItem(...) getItem(...) removeItem(...)</pre> <p>These methods allow an application to communicate with another application.</p> <p>If this field is set to <code>false</code>, or if this field is not present, the methods above will not be usable by the application. If the application attempts to make use of these methods, a callback error response will be returned to the Application</p> <p>If this field is set to <code>true</code>, the methods will be available for use by the application</p> <p>Note If a developer wishes to store data for use by another application, they must share their application's namespace code with the developers of the application that will read the data. For further details see the [PNG-IC-0203].</p>
appsList-AN-D	An Application Object does not contain a valid <i>allowInterAppStorage</i> field	–	<p>The Framework will treat the absence of a valid <i>allowInterAppStorage</i> field as the equivalent to:</p> <pre>allowInterAppStorage = false</pre>

6.2. appsList.json structure

This section describes the data structure of the appsList.json file.

6.2.1. Root

The following fields are required, unless otherwise stated.

Field	Description	Required / Type	Example
appList	The root object	Object	
legals...	A list of legal information that can be associated with each application	Legals Object	
applications...	One or more ATSC 3.0 Broadcast Application definitions	Applications Object	

6.2.2. Legals Object

The following fields are required, unless otherwise stated.

Field	Description	Required / Type	Example
<LegalID value>	An identifier used by applications to link to the correct legal information. This identifier is case sensitive and must be of the form: <code>^[a-zA-Z0-9]*\$</code>	Object	custom
match-keys	A set of key value pairs that can be inserted into the terms-of-service and privacy-policy files	Object	
name	The legal entity name to be presented to viewers	String	Example Ltd
address	The contact postal address for any viewer legal queries	String	123 Example St, ExampleTown, 37160
email	The contact email address for any viewer legal queries	String	info@example.com
date	The most recent date that the terms of service and privacy policy have been published	String	Tuesday, December 08, 2022
terms-of-service	The local file path to the terms of service html file	String (Path)	/london/Station-2/common-app/tos/terms-of-service.html
privacy-policy	The local file path to the privacy policy html file	String (Path)	/london/Station-2/common-app/tos/privacy-policy.html

6.2.3. Terms of service and privacy policy files

The files referenced by the terms-of-service and privacy-policy fields must be text files that include HTML markup. These files will be inserted within existing HTML content, so no <HTML> or <Body> tags are required. The example privacy-policy file shown later in this document shows the complete tagging structure.

Any of the fields within the match-keys object can be included using curly braces “{}” to automatically include the associated text.

The match-keys object can contain any key:<String> data.

If the terms-of-service or privacy-policy files contain a reference to a key that does not exist within match-keys, that reference (including curly braces) will not be substituted.

The truncated example of a privacy-policy file below gives an example making use of the “custom” legal's object above. This includes an example fault where `{version}` has not been added to the match-keys object:

Example privacy-policy file	Viewer-facing result
<pre><p>Broadcaster: {name}</p> <p>Effective Date: {date}</p> <p>Privacy questions: {email}</p> <p>This Privacy Policy (version {version}) sets forth the Broadcaster's practices for handling the information we collect from or about you on the NextGen television app (the "service").</pre>	<p>Broadcaster: COMPANY 12</p> <p>Effective Date: Tuesday, December 08, 2020</p> <p>Privacy questions: contact@domain.net</p> <p>This Privacy Policy (version {version}) sets forth the Broadcaster's practices for handling the information we collect from or about you on the NextGen television app (the "service").</p>

6.2.4. Applications Object

The following fields are required, unless otherwise stated.

Field	Description	Required / Type	Example
<code><appName Id></code>	The application identifier This identifier is case sensitive and must be of the form: <code>^[a-zA-Z0-9]*\$</code>	Object	<code>{"STATION-1" : {...}}</code>
<code>appRoot</code>	The local directory path to the base of the application	String (Path)	<code>/run3tv-common/apps</code>
<code>appEntry</code>	The filename of the entry HTML file (from the <code>appRoot</code> location)	String (Path)	<code>/index.html</code>
<code>appVersion</code>	The version of the application. This is presented to viewers in the settings section of the application (where present) and also provided along with any analytics data. This value can take any string form. The Framework does not require this version number to increment	String	<code>1.0.0</code>
<code>appID</code>	A unique application identifier string. For applications written for versions of the Framework prior to v2.0: The Google Analytics ID. For Framework v2.0 applications: Any unique identifier string	String	<code>com.run3tv.fieldtest</code>

Field	Description	Required / Type	Example
<code>properties</code>	The configuration properties for the application. Prior to v2.0 of the Framework, this value was an optional set of key-value pairs. v2.0 of the Framework is an optional string, referencing a config.json file	Optional Properties Object/String	
<code>allowInterAppStorage</code>	This flag enables or disables inter-app data sharing. Applications can read and write data using: <code>setItem(...)</code> <code>getItem(...)</code> <code>removeItem(...)</code> For further details see the [PNG-IC-0203]	Optional Boolean (default: False)	true

6.2.5. Application Properties

Applications can be configured in various .json files using the `properties` key.

Functionality differs between versions of the framework and the example applications provided.

Previous versions of the Framework required a number of configuration key/value pairs, as documented in Annex B of this document.

The current version of the Framework (v2.0 up) additionally accepts a String file path reference to a **config.json** file. This file contains all configuration options.

Note For further information on configuring the properties of applications, please refer to the relevant application's documentation. An example of the Q-Bar's configuration can be found in [ANNEX D - Examples of application configurations](#).

The following examples are equivalent to each other:

Example 1: Inline configuration	
bridge.json	<pre>{ "connection": "bband", "bridge": { "tag:yottamedialabs.com,2022:globalServiceID/2": { "bbandappList": "/london/appsList.json", "appDefault": { "appName": "TRIGGER-1", "properties": { "ctaLabel": "Press <baAppearLabel> for Sport News", "debug": { "showVersion" : true } } }, "serviceInfo": { "broadcaster": "Yotta", "network": "Run3TV", "stationDMACode": 999, "callSign": "RYML-2" } } } }</pre>
config.json	Not present

Example 2: Referenced configuration**bridge.json**

```
{
  "connection": "bband",
  "bridge": {
    "tag:yottamedialabs.com,2022:globalServiceID/2": {
      "bbandappList": "/london/appsList.json",
      "appDefault": {
        "appName": "TRIGGER-1",
        "properties": "config.json"
      },
      "serviceInfo": {
        "broadcaster": "Yotta",
        "network": "Run3TV",
        "stationDMACode": 999,
        "callSign": "RYML-2"
      }
    }
  }
}
```

config.json

```
{
  "ctaLabel": "Press <baAppearLabel> for Sport News",
  "debug": { "showVersion" : true }
}
```

7. Application scheduling for a channel (appSchedule.json)

It is possible to schedule applications using **appSchedule.json** files referenced in the **bridge.json** file.

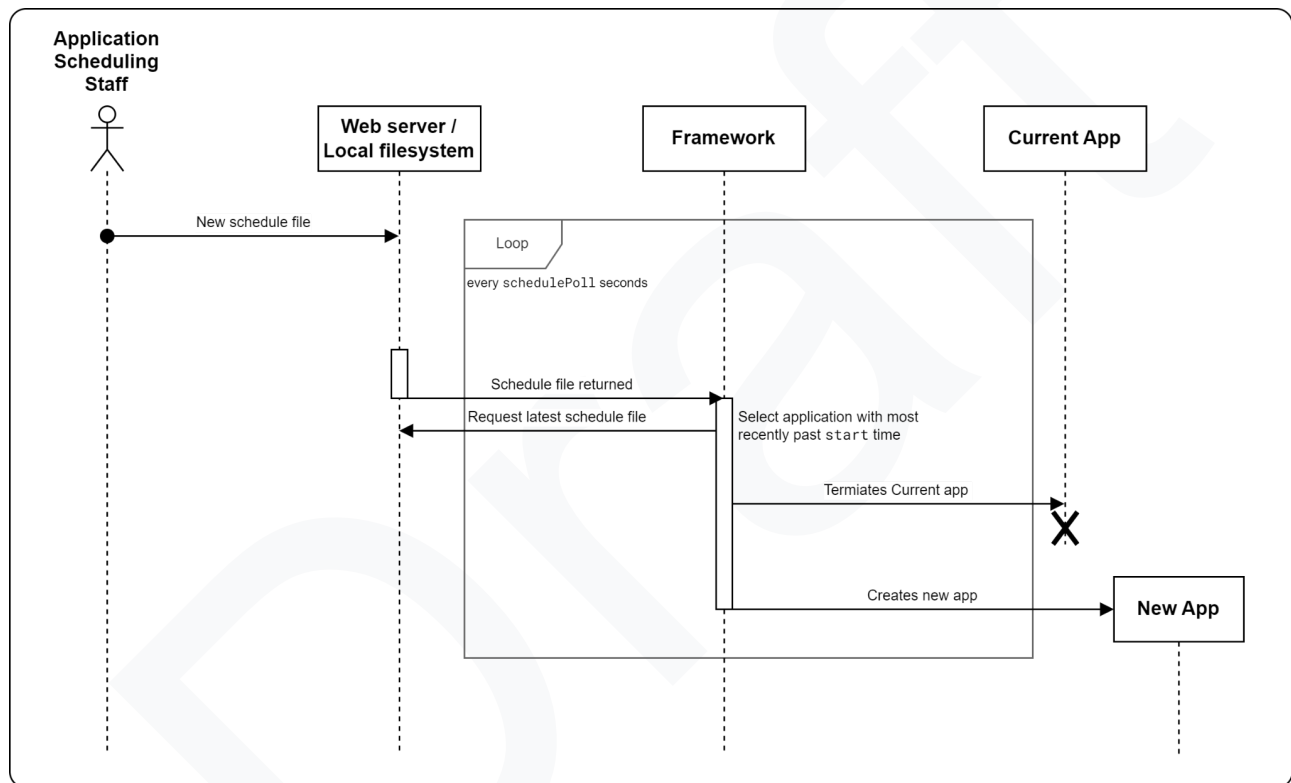
appSchedule.json files can be:

- referenced using the **bbandappSchedule** and/or **bcastappSchedule** fields in the **bridge.json** file
- Stored locally within the Framework or hosted on a web server that is accessible to the Framework
- Updated dynamically to reflect new schedules

7.1. Scheduling flow

appSchedule.json files contain a schedule with one or more events. These events contain a start time and the application to make available to viewers at that time. The application is selected from the collection of applications within the **appsList.json** file using the **appName** field

If no application is to be available for a period, an event should be created that references the “blank” application. This “blank” application must also be included in the **appsList.json** file.



Application scheduling staff can update the **appSchedule.json** file as new schedules are known. This update process may involve:

- Using a CMS to create and publish new schedules
- Using an automated scheduling system to automatically publish new schedules
- Manually crafting and uploading new schedules

There is no limit to the number of scheduled events that can be included in an **appSchedule.json** file, however large files will result in larger downloads and increased processing for receivers. It is recommended to keep **appSchedule.json** files below 100 kilobytes in size.

If additional scheduled events are required in the future, the Framework allows for **appSchedule.json** files to be regularly updated, with old events removed and new future events added. The Framework can be configured to regularly poll for these updates.

A simplified example **appSchedule.json** file is shown below.

```

1      {
2        "appSchedules": {
3          "schedulePoll": 10,
4          "graceTimeout": 20,
5          "schedule": {
6            "event1": {
7              "start": 1697911200,
8              "appName": "cta-ba",
9              "properties": {
10             }
11            },
12            "event2": {
13              "start": 1697914800,
14              "appName": "trigger-1",
15              "properties": {
16             }
17            },
18            "event3": {
19              "start": 1697915700,
20              "appName": "blank",
21              "properties": {
22             }
23            }
24          }
25        }
26      }

```

This **appSchedule.json**:

- Configures the Framework to monitor the appSchedule.json file for updates every 10 seconds, using the `schedulePoll` field (line 3)
- Sets any application currently in use by viewers to remain available after any application schedule change for 20 seconds, using the `graceTimeout` field (line 4)
- Contains three scheduled events, each with their own start time and associated application.

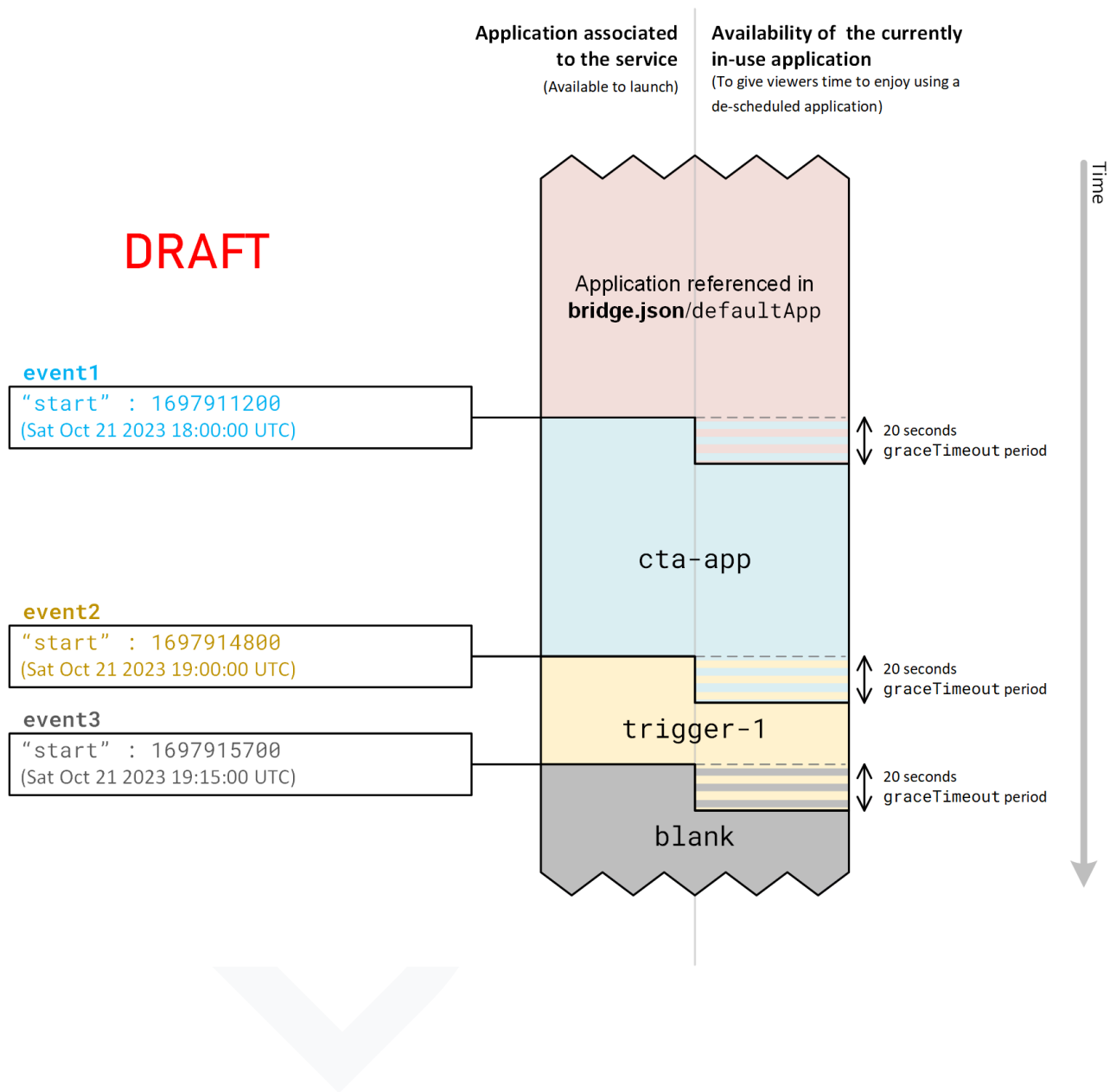
Each schedule contains a start time (supplied as a Unix Epoch value) and an application (referenced using the `appName` field) to make available at this time.

Note Events must be listed in sequential order.

All times are in UTC.

To provide a frictionless experience to viewers that are interacting with applications that are de-scheduled and replaced with a new application, the `gracePeriod` field can be used to set the number of seconds after an application is de-scheduled (but is still being actively used by the viewer) until the application is forcibly terminated.

The example **appSchedule.json** file above can be summarized as shown below:



7.2. Scheduling scenarios

The table below describes common scheduling scenarios

Scenario	bridge.json	appSchedule.json	Description
No appSchedule.json provided in bbandappSchedule	Default app is set. bbandappSchedule field is not present	<i>Not present</i>	The defaultApp is automatically associated with the service
appSchedule.json is provided in bbandappSchedule . All schedule events are in the future	Default app is set. bbandappSchedule field is present	All scheduled events are set in the future	The defaultApp is automatically associated with the service. When the earliest event's time point arrives, the next time the Framework polls appSchedule.json , that event will be considered active. That event's application will be associated with the service
appSchedule.json is provided in bbandappSchedule . At least one scheduled event is in the past		At least one scheduled event is set in the past	When the Framework polls appSchedule.json , the event in the most recent past will be considered active. That event's application will be associated with the service

7.3. Precision

The **appSchedule.json** file offers to-the-second precision in timing, however there is a delay between this scheduled time and the new application becoming available on receivers.

This delay is dependent on:

- The polling time of the **appSchedule.json** file
- The time taken for the Framework to unload the existing application
- The time taken for the Framework to load the new application with all its dependencies.

Application scheduling staff should perform tests in their market to determine the average delay times and adjust the scheduled timings accordingly.

Note The development roadmap for the Framework includes extending the application scheduling functionality to allow scheduling of applications using event triggering in the ATSC 3.0 DASH manifest file.

This will provide more precise application scheduling.

7.4. Invalid appSchedule.json files

If, when the Framework first starts up, the **appSchedule.json** file is invalid or not well formed, the Framework will consider the service's schedule for the service invalid and will not attempt to reload **appSchedule.json**. Instead, the Framework will:

- Present an error to the console log; and,
- Associate the default app with the service, as defined in the **bridge.json** defaultApp object.

If the Framework has previously parsed a successful appSchedule.json file whilst running and then detects an update to the appSchedule.json that is invalid or not well formed, the Framework will:

- Present an error to the console log; and,
- Retain the existing schedule and polling interval.

The Framework will continue to retain the previous schedule and poll until it can access an updated, valid, well formed **appSchedule.json** file.

- appSchedule.json file is updated with invalid content, the Framework will retain the previous schedule and continue to poll for updates until a valid replacement schedule is provided.

The following sections describe the structure of **appSchedule.json** in detail.

7.5. Business rules for appSchedule.json

7.5.1. Structure of appSchedule.json

Req	Scenario	Impact	Description
AppsSch-St-1	The appSchedule.json file is present, valid, and well formed	—	The Framework shall parse this file and configure the schedule for the service as defined
AppsSch-St-2	The appSchedule.json file is either missing, mis-named or mislocated	Default application presented	The Framework shall present an error in the console log. The default application (as defined in bridge.json) will be associated with the service rather than the intended schedule
AppsSch-St-3	The appSchedule.json file is present, but does not contain valid, well formed json data	Default application presented	The Framework shall present an error in the console log. The default application (as defined in bridge.json) will be associated with the service rather than the intended schedule
AppsSch-St-4	The appSchedule.json file is present, valid and well formed, but contains additional data fields	—	The Framework shall parse this file and configure the schedule for the service as defined. It shall ignore any fields that it does not expect

7.5.2. Updates to appSchedule.json

Req	Scenario	Impact	Description
AppsSch-Up-1	The appSchedule.json file is updated to remove the event that the Framework is currently presenting to the viewer	Variable	The Framework will parse the new appSchedule.json file and select the most appropriate application to load based on the Scheduling flow
AppsSch-Up-2	The appSchedule.json file is updated to modify or remove an event that the Framework is not currently presenting to the viewer	Variable	The Framework will parse the new appSchedule.json file and select the most appropriate application to load based on the Scheduling flow
AppsSch-Up-3	The appSchedule.json file is updated with the same schedule but different configuration for the current event	Application config will not update	The Framework will parse the new appSchedule.json file and determine that the correct application is already running. The Framework will not relaunch the application with the new configuration, nor will it inform the application to update its configuration
AppsSch-Up-4	The appSchedule.json file is updated to modify the appName within the existing event that the Framework is currently presenting to the viewer	New application launched	The Framework will parse the new appSchedule.json file and determine that a new application must be launched. The new application will be launched and the associated properties will be passed to it
AppsSch-Up-5	The appSchedule.json file is updated to change the schedulePoll value to a new positive integer	Polling time updated	The Framework will parse the new appSchedule.json file and schedule the next reload of this file to the value set in the schedulePoll field
AppsSch-Up-6	The appSchedule.json file is updated to change the schedulePoll value to zero or a negative integer	New schedule will be ignored	The Framework will consider the new appSchedule.json file invalid. The Framework shall retain the previous event schedule and set a new reload timer using the previous valid appSchedule.json's schedulePoll value
AppsSch-Up-7	The appSchedule.json file is updated such that the same application is scheduled, however the graceTimeout value has changed to a new positive integer or zero	Grace timeout will be updated only for a new app	The Framework will not inform the currently running application of the new graceTimeout value. Any future application will use the new graceTimeout value
AppsSch-Up-8	The appSchedule.json file is updated such that the same application is scheduled, however the graceTimeout value has changed to a negative integer	New schedule will be ignored	The Framework will consider the new appSchedule.json file invalid. The Framework shall retain the previous event schedule and set a new reload timer using the previous valid appSchedule.json's schedulePoll value
AppsSch-Up-9	The Framework attempts to reload the appSchedule.json file, however that file is either: <ul style="list-style-type: none"> Updated with invalid or non-well formed content; or, Unreachable by the Framework 	Various	<p>If invalid or non well formed content is present in the appSchedule.json file when the Framework first launches:</p> <ul style="list-style-type: none"> The Framework will mark the file as invalid and will associate the defaultApp (as defined in bridge.json). No further attempts to read the appSchedule.json file will occur. The framework shall present an error in the console log. <p>If this is present after the Framework has successfully parsed at least one previous version of appSchedule.json file, the Framework shall:</p> <ul style="list-style-type: none"> discard this file. present an error in the console log. retain the previous schedule and start a new schedulePoll timer using the previous schedulePoll value

7.5.3. Top level and secondary level appSchedule.json fields

Req	Scenario	Impact	Description
AppsSch-TL-1	The appSchedules field is present and contains valid, well formed data	None	The Framework shall parse this file and configure itself for the schedule as defined
AppsSch-TL-2	The appSchedules / schedulePoll field is present and contains a positive integer	None	<p>Upon:</p> <ul style="list-style-type: none"> Completion of parsing the current appSchedule.json file; and, Completion of associating any application scheduled in the current appSchedule.json file to the service; and, Completion of any other tasks relating to appSchedule.json, <p>...the Framework shall set a timer to the number of seconds defined in this field.</p> <p>When the timer fires, the Framework shall reload the appSchedule.json file, reparse it and if appropriate (based on the new schedule in the appSchedule.json file), associate a new application to the service, and set a new schedulePoll timer</p> <p>Note The schedulePoll value directly affects the accuracy of application scheduling. The Framework only parses the appSchedule.json file and associates new applications with a service once every <schedulePoll> seconds. If high precision application scheduling, this value should be a low number of seconds.</p>
AppsSch-TL-3	The appSchedules / graceTimeout field is present and contains either a positive integer or zero	None	<p>The Framework shall set this as the maximum amount of time that a viewer may remain using an application that has been de-scheduled.</p> <p>If this value is set to zero, any viewer using an application that has just been de-scheduled and replaced with a different application will be immediately kicked out of the application and returned to full screen TV</p> <p>If it is editorially desirable to allow viewers to remain interacting with a de-scheduled application indefinitely, this field should be set to a very large number, noting the JavaScript limitation of Number . MAX_SAFE_INTEGER</p>
AppsSch-TL-4	The appSchedules / schedule field is present and contains valid, well formed content	None	The Framework shall parse this file and configure itself for the schedule as defined

7.5.4. Schedule Object entries in appSchedule.json

Req	Scenario	Impact	Description
AppsSch-SO-1	The <event ID> field is present, a unique identifier and contains valid, well formed data	None	The Framework shall parse this file and configure itself for the scheduled event as defined
AppsSch-SO-2	The <event ID> field is present and is a duplicate of another <event id>	Default application presented	<p>The Framework shall present an error in the console log.</p> <p>The default application (as defined in bridge.json) will be associated with the service rather than the intended schedule</p>
AppsSch-SO-3	Schedule Objects are listed in chronological order (based on their start fields)	None	The Framework shall parse this file and configure itself for the schedule as defined by the Schedule Objects
AppsSch-SO-4	Schedule Objects are listed out of chronological order (based on their start fields)	Undefined	The Framework shall parse the file and select the first event that is closest to the current time, as set by the receiver

Req	Scenario	Impact	Description
AppsSch-SO-5	<p>The start field is present and contains a unix epoch time value that is either:</p> <ul style="list-style-type: none"> Now <p>Or:</p> <ul style="list-style-type: none"> Is in the past; and, Is closest in time to Now than any other Schedule Object's start field <p>Note "Now" is the point in time when the appSchedule.json file is parsed.</p>	None	<p>When the Framework parses the appSchedule.json file, it will associate the application reference in appName field and configured using the properties field of the Schedule Object.</p> <p>Any other Schedule Object will be ignored, as they either are events from the past or future</p>
AppsSch-SO-6	The start field is present however shares the same unix epoch as another Schedule Object's start field	None	<p>When the Framework parses the appSchedule.json file, it will mark both schedules as invalid and shall present an error in the console log.</p> <p>Neither application referenced in these Schedule Objects will not be associated with a service.</p> <p>Any other Schedule Objects will be parsed as normal</p>
AppsSch-SO-7	The appName field is present and its value exists in the service's appsList.json file	None	The Framework will consider this a viable Schedule Object and will associate the application with the service as defined above
AppsSch-SO-8	The appName field is present and its value exists in the service's appsList.json file	Schedule Object skipped	<p>The Framework will not consider this a viable Schedule Object and will not use it to associate an application with the service.</p> <p>The Framework shall present an error in the console log</p>
AppsSch-SO-9	The properties field is present and is either a valid object or references a config.json file accessible to the receiver	None	The Framework will consider this a viable Schedule Object and will associate the application with the service as defined above. When the application is launched, the Framework will provide the configuration as defined in the properties field to that application
AppsSch-SO-A	The properties field is not present	None	The Framework will consider this a viable Schedule Object and will associate the application with the service as defined above. When the application is launched, the Framework will provide the configuration as defined in appsList.json to that application
AppsSch-SO-B	The properties field is present and is either an invalid object or references a config.json file that is not accessible to the receiver	Schedule Object skipped	<p>The Framework will not consider this a viable Schedule Object and will not use it to associate an application with the service.</p> <p>The Framework shall present an error in the console log</p>

7.5.5. Viewer experience on a service with a schedule

Req	Scenario	Impact	Description
AppsSch-VX-1	The viewer is watching TV and the Framework detects that a new event time point has occurred. This new event has a different application associated with it	—	<p>The Framework tears down the current (previously scheduled) application and launches the new application (if available).</p> <p>If the application automatically presents a call to action message, this will be presented to the viewer</p>

Req	Scenario	Impact	Description
AppsSch-VX-2	The viewer is interacting with an application and the Framework detects that a new event time point has occurred. This new event has a different application associated with it	–	The Framework will hold off tearing down the current (previously scheduled) application and launching the new application until either: <ul style="list-style-type: none"> • The graceTimeout period has elapsed; or, • The viewer exits the current application
AppsSch-VX-3	The viewer is interacting with an application and the Framework detects that a new event time point has occurred. This new event has a different application associated with it. The viewer continues to interact with the application beyond the gracePeriod time	–	The Framework will tear down the current application and return the viewer to full screen TV. The Framework will then launch the new application. If the application automatically presents a call to action message, this will be presented to the viewer
AppsSch-VX-3	The viewer is watching TV and the Framework detects that a new event time point has occurred. This new event has a the same application as currently available associated with it	Various	The Framework will detect the new event, however as the application is already loaded, the Framework will not reload it. This means that if the event's properties field contains a different configuration to the current application, this configuration will not be acted upon

7.6. appSchedule.json structure

This section describes the data structure of **appSchedule.json**.

7.6.1. Root

The following fields are required, unless otherwise stated.

Field	Description	Required / Type	Example
appSchedules		Object	
<code>schedulePoll</code>	The polling interval (in seconds) for the Framework to check for updates to this file	Integer	10
<code>graceTimeout</code>	The amount of time (in seconds) to allow viewers still interacting with the previously scheduled application to remain in that application before it is forcibly closed. If the application is hidden, or the trigger is present, no grace period is used and the application is terminated immediately	Integer	30
<code>schedule... {1+}</code>	One or more events to schedule different applications	Schedule Object	

7.6.2. Schedule Object

The following fields are required, unless otherwise stated.


Field	Description	Required / Type	Example
<event ID>	A unique identifier for the event	Object	Event
<code>start</code>	The time point to start the new application Note All times are in UTC	Unix Epoch	1654761240
<code>appName</code>	The name of the application (as defined in the appsList.json file for the service) If this name does not exist in the appsList.json file, the default application as defined in the <code>appDefault</code> field in the bridge.json file	String (AppName)	cta-ba
<code>properties</code>	Any properties required by the application. This value will completely replace any properties value set in appsList.json . These properties can be provided in-line, or by providing a reference to a config.json file. If the properties as defined in appsList.json are to be used, this field should not be included	Optional Object / Local File Reference	{ ctaLabel: "PRESS <baAppearLabel> for the News" }

8. Substitution with Replaceable Field Tags

To enable application developers to make use of information that is either deployment-specific or only known at runtime, the Framework offers a number of substitution tokens referred to as Replaceable Field Tags. These substitutions can be used throughout the application and its configuration. Common uses include:

- Providing receiver-specific viewer-facing strings
- Generating receiver-specific urls to feeds

All available Replaceable Field Tags are listed below:

Replaceable Field Tag	Description	Type	Example replacement values	Example usage
<advertisingId>	The receiver's advertising Id. If the receiver does not support this feature, an empty string will be substituted. If the viewer is explicitly not authorized use of this Id, a valid ID URN with all zeros shall be returned For further details, see [A344-2023-02] § 9.12	String (UUID)	urn:uuid:3f614541-14bb-4816-a868-77b3d5223262	
<appId>	The appId as set by the appList.json 's appId ¹ field for the application. If this value is not provided, appName shall be returned instead	String	com.run3tv.fieldtest	
<baAppearLabel>	The BA Appear Label. This is the text printed on the remote control unit's BA Appear button. For further information, see the <i>deviceInfoProperties</i> JSON-RPC message in [A344-2023-02]	String	OK ENTER SELECT	"Press <baAppearLabel> to open" ↓ "Press ENTER to open"
<baAppearImage>	The BA Appear Label. This is the text printed on the remote control unit's BA Appear button. For further information, see the <i>deviceInfoProperties</i> JSON-RPC message in [A344-2023-02]	String (Inline encoded image)		
<baseURI>	The base URI of the Framework on the receiver. This may be unique per receiver	String (URI)		
<callsign>	The callsign of the station, as set in the bridge.json file	String	KACL KFLW-TV	
<countryFipsCode>	The two character FIPS country code of the receiver. If this value is not available, an empty string will be substituted	String	US KS	

¹ For the avoidance of doubt, this data is taken from the appList/applications/<appName Id>/appId field. The appList/applications/<appName Id>/properties/appId is not used for this purpose.

Replaceable Field Tag	Description	Type	Example replacement values	Example usage																								
<date:{Datecode}>	<div>The current date as determined by the receiver. Datecodes can be generated from one or more of the following tokens:</div> <table><thead><tr><th>Datecode token</th><th>Example</th></tr></thead><tbody><tr><td>mmm</td><td>mar</td></tr><tr><td>Mmm</td><td>Mar</td></tr><tr><td>MMM</td><td>MAR</td></tr><tr><td>mm</td><td>03</td></tr><tr><td>MM</td><td>03</td></tr><tr><td>dd</td><td>01</td></tr><tr><td>DD</td><td>01</td></tr><tr><td>yyyy</td><td>2023</td></tr><tr><td>YYYY</td><td>2023</td></tr><tr><td>yy</td><td>23</td></tr><tr><td>YY</td><td>23</td></tr></tbody></table> <div>Datecode tokens can be combined, as shown in the examples column</div>	Datecode token	Example	mmm	mar	Mmm	Mar	MMM	MAR	mm	03	MM	03	dd	01	DD	01	yyyy	2023	YYYY	2023	yy	23	YY	23	String	<div>dependant on the receiver's current date</div>	<div><date:mm-dd-yy> ↓ 03-01-23</div> <div><date:mmm/dd/yyyy> ↓ mar-01-2023</div>
Datecode token	Example																											
mmm	mar																											
Mmm	Mar																											
MMM	MAR																											
mm	03																											
MM	03																											
dd	01																											
DD	01																											
yyyy	2023																											
YYYY	2023																											
yy	23																											
YY	23																											
<ga-id>	The Google Analytics identifier, as configured in the application's properties/ga-id field	String	G-MXP1YLWM3C																									
<gsid>	<div>The Global Service ID of the currently tuned service</div> <div>Note If the application has tuned to a different service, this GSID may be different to the GSID as defined in the bridge.json file</div>	String	tag:yottamedialabs.com,2022:globalServiceID/2																									
<language>	The two character BCP 47 language code of the receiver. For example, this can be used within configuration files to direct the receiver to the correct language variant for a feed	String	en es	<div>https://example.com/feeds-<Language>.atom ↓ https://example.com/feeds-es.atom</div>																								
<location>	The county that the receiver is located in. If this value is not available, an empty string will be substituted.	String	Orange Queens Miami-Dade																									
<manufacturer>	The receiver manufacturer	String	Panasonic Samsung																									
<model>	The year and model of the receiver	String	QBQS95 Sony-BRAVIA-VH1																									
<sessionId>	The unique session id of the framework	String (UUID)	3ce77ae1-2290-4964-8904-ce26bfe914bf																									

Replaceable Field Tag	Description	Type	Example replacement values	Example usage
<zipCode>	The ZIP code of the receiver, as entered by the viewer. If this value is not available, an empty string will be substituted.	String	02108 34145	

For example, the <baAppearLabel> token enables applications to use the same button text as appears on the viewer's remote control.

The example below demonstrates how a viewer with an "OK" button on their remote control unit would see the call to action.

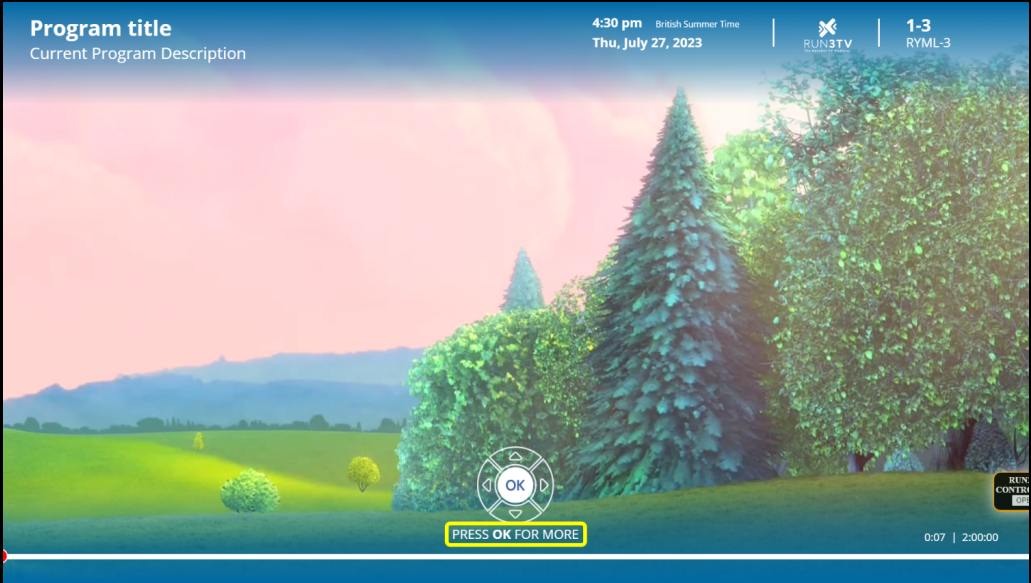
Call to Action string with <baAppearLabel> Replaceable Field Tag

Configuration

```

...
"ctaLabel": "PRESS <baAppearLabel> FOR MORE",
...

```

On-screen result


The screenshot shows a TV interface with a landscape background. At the top, it displays 'Program title' and 'Current Program Description'. On the right, it shows the time '4:30 pm', 'British Summer Time', the date 'Thu, July 27, 2023', the 'RUN3TV' logo, and the channel '1-3 RYML-3'. In the center, there is a large 'OK' button with a circular arrow around it. Below the button, a yellow banner reads 'PRESS OK FOR MORE'. In the bottom right corner, there is a small 'RUN3TV' logo and a timer '0:07 | 2:00:00'.

9. Multi language string object

It is possible to supply multiple language text strings for various fields within config.json. This is achieved by creating an object of key :<String> pairs, where the key is an BCP 47 language code.

An example of a title field with two languages is shown below. The Framework selects the most appropriate language string to present to the viewer, based on the viewer's configuration choice.

Multi language string object: Example of a title field containing two languages

```
...
  "title": {
    "en": "Local test streams",
    "es": "Flujos de prueba locales"
  }
...
```

ANNEX A - bridge.json example

```
{
  "connection": "bband",
  "bridge": {
    "tag:yottamedialabs.com,2022:globalServiceID/2": {
      "bbandapplist": "/london/appsList.json",
      "appDefault": {
        "appName": "field-test",
        "properties": {}
      },
      "serviceInfo": {
        "broadcaster": "Yotta",
        "network": "Run3TV",
        "stationDMACode": 999,
        "callSign": "RYML-2"
      }
    },
    "tag:yottamedialabs.com,2022:globalServiceID/3": {
      "bcastapplist": "/london/appsList.json",
      "bcastappSchedule": "/london/Station-1/schedule/appSchedule.json",
      "appDefault": {
        "appName": "cta-ba",
        "properties": {}
      },
      "serviceInfo": {
        "broadcaster": "Yotta",
        "network": "Run3TV",
        "stationDMACode": 999,
        "callSign": "RYML-3"
      }
    },
    "tag:yottamedialabs.com,2022:globalServiceID/4": {
      "bbandapplist": "/london/appsList.json",
      "bbandappSchedule": "/london/Station-2/schedule/appSchedule-250222.json",
      "appDefault": {
        "appName": "trigger-2",
        "properties": {}
      },
      "serviceInfo": {
        "broadcaster": "Yotta",
        "network": "Run3TV",
        "stationDMACode": 999,
        "callSign": "RYML-4"
      }
    },
    "tag:yottamedialabs.com,2022:globalServiceID/5": {
      "bbandapplist": "/london/appsList.json",
      "appDefault": {
        "appName": "leadgen-cta",
        "properties": {}
      },
      "serviceInfo": {
        "broadcaster": "Yotta",
        "network": "Run3TV",
        "stationDMACode": 999,
        "callSign": "RYML-5"
      }
    }
  }
}
```

```
"tag:yottamedialabs.com,2022:globalServiceID/6": {
  "bbandappList": "/london/appsList.json",
  "appDefault": {
    "appName": "dai-google-pre",
    "properties": {}
  },
  "serviceInfo": {
    "broadcaster": "Yotta",
    "network": "Run3TV",
    "stationDMACode": 999,
    "callSign": "RYML-6"
  }
},
"tag:yottamedialabs.com,2022:globalServiceID/7": {
  "bbandappList": "/london/appsList.json",
  "appDefault": {
    "appName": "lottie",
    "properties": {}
  },
  "serviceInfo": {
    "broadcaster": "Yotta",
    "network": "Run3TV",
    "stationDMACode": 999,
    "callSign": "RYML-7"
  }
}
}
```

ANNEX B - appsList.json example

The **appsList.json** example below describes two variants of the same Q-Bar application with unique configuration settings for news and regular programming. These configuration settings may prioritize news feeds and general entertainment feeds respectively.

An **appSchedule.json** file could be set to switch between the two applications at the appropriate times of day.

appsList.json Example 1

```

1  {
2    "appList": {
3      "legals": {
4        "default": {
5          "match-keys": {
6            "name": "COMPANY 12",
7            "address": "Company Address No. 12",
8            "email": "contact@domain.net",
9            "date": "Tuesday, December 08, 2020"
10         },
11         "terms-of-service": "/run3tv-common/tos/terms-of-service.html",
12         "privacy-policy": "/run3tv-common/tos/privacy-policy.html"
13       }
14     },
15     "applications": {
16       "STATION-1-regular": {
17         "appRoot": "/run3tv-common/apps/common-app",
18         "appEntry": "/index.html",
19         "appVersion": "1.0.0",
20         "appId": "com.run3tv-common/q-bar/regular",
21         "properties": "https://example.com/station1Regular/config.json",
22         "allowInterAppStorage": true
23       },
24       "STATION-1-news": {
25         "appRoot": "/run3tv-common/apps/common-app",
26         "appEntry": "/index.html",
27         "appVersion": "1.0.0",
28         "appId": "com.run3tv-common/q-bar/newshour",
29         "properties": "https://example.com/station1News/config.json",
30         "allowInterAppStorage": true
31       }
32     }
33   }
34 }

```

ANNEX C - appSchedule.json example

The appSchedule.json example below matches the appsList.json in Annex B that schedules the same Q-Bar application with different configurations.

The schedulePoll value is set to 600 seconds, as it is expected that the appSchedule.json file will be updated well in advance of any new schedules, and that the schedule is well known in advance.

Note If the broadcast service runs a dynamic schedule, where breaking news may interrupt regular programming, and the broadcaster is able to provide updated **appSchedule.json** files based on this dynamic schedule, it may be appropriate to choose a shorter schedulePoll value.

The graceTimeout value is set to a very long time period, allowing viewers already using an application to remain almost indefinitely in that application, despite a schedule change occurring.

appSchedule.json Example 1

```

1  {
2    "appSchedules": {
3      "schedulePoll": 60,
4      "graceTimeout": 9999999,
5      "schedule": {
6        "event1": {
7          "start": 1654761240,
8          "appName": "cta-ba"
9        },
10       "event2": {
11         "start": 1655459340,
12         "appName": "STATION-1-news"
13       },
14       "event3": {
15         "start": 1655459340,
16         "appName": "STATION-1-regular"
17       }
18     }
19   }
20 }
```

ANNEX D - Examples of application configurations

Each RUN3TV application may have a **config.json** file that is used to set its various properties and feature flags or that the configurations are passed via the `properties` field directly in the json files.

Please refer to each application's documentation for specific configuration details.

A simplified example **appsList.json** and **config.json** file for the Q-Bar application is shown below.

appsList.json

```

1  {
2    "appList": {
3      "legals": {
4        "default": {
5          "match-keys": {
6            "name": "COMPANY 12",
7            "address": "Company Address No. 12",
8            "email": "contact@domain.net",
9            "date": "Tuesday, December 08, 2020"
10         },
11         "terms-of-service": "/run3tv-common/tos/terms-of-service.html",
12         "privacy-policy": "/run3tv-common/tos/privacy-policy.html"
13       },
14       "custom": {
15         "match-keys": {
16           "company": "COMPANY 21",
17           "address": "Company Address No. 21"
18         },
19         "terms-of-service": "/london/Station-2/common-app/tos/terms-of-service.html",
20         "privacy-policy": "/london/Station-2/common-app/tos/privacy-policy.html"
21       }
22     },
23     "applications": {
24       "STATION-1": {
25         "appRoot": "/run3tv-common/apps/common-app",
26         "appEntry": "/index.html",
27         "appVersion": "1.0.0",
28         "appId": "G-B1Q1EGVSX",
29         "properties": "/run3tv-common/config.json"
30       }
31     }
32   }
33 }
34 }
```

config.json

```

1  {
2      "ga-id": "G-MXP1YLWM3C",
3      "hidden": false,
4      "timeouts": {
5          "trigger": 60,
6          "inactivity": 90,
7          "player": 5,
8          "skip": 400
9      },
10     "player": {
11         "showCaption": true
12     },
13     "debug": {"showVersion": false, "enableConsole": false},
14     "style": {
15         "default": "/london/Station-3/q-bar/css/style.css",
16         "thumbnail": "/london/Station-3/q-bar/thumbnail",
17         ":root": {
18             "--font-family": "\"Open Sans\", sans-serif",
19             "--title-font-size": "24px",
20             "--description-font-size": "20px",
21             "--bar-bg-color": "#008CDE",
22             "--font-color": "#FFF",
23             "--focus-font-color": "#008CDE",
24             "--focus-bg-color": "#FFF",
25             "--focus-font-color-sub1": "red",
26             "--focus-font-color-sub2": "black",
27             "--focus-font-color-sub3": "#FDC109",
28             "--vbar-bgcolor-end": "#0069A6",
29             "--gradient-color-start": "#008CDE80",
30             "--gradient-color-end": "#008CDE10",
31             "--trigger-vbar-bg-color": "#FFF",
32             "--trigger-font-color": "#FFF",
33             "--trigger-bg-top": "linear-gradient(180deg, #006AA6 0%, #0069a67e 60%, transparent 100%)",
34             "--trigger-bg-bottom": "linear-gradient(0deg, #006AA6 0%, #0069a67e 60%, transparent 100%)",
35             "--ticker-bg-color": "#0C3C66",
36             "--opacity": 0.4
37         }
38     },
39     "ctaLabel": {"en": "PRESS <strong><baAppearLabel></strong> FOR MORE", "es": "PRESIONE
40     <strong><baAppearLabel></strong> PARA MAS"},
41     "header": {"id": "SRV1", "name": "ID-1", "logo": "/london/Station-3/q-bar/img/channel.svg"},
42     "feeds": [
43         {"feedType": "mrss", "title": {"en": "Local test streams", "es": "Flujos de prueba locales"},
44         "url": "/london/Station-3/q-bar/dynamic/test.atom"},
45         {"feedType": "mrss-json", "title": "Local HLS Feed", "url":
46         "/london/Station-3/q-bar/dynamic/a3fa-hls-feed.json"},
47         {"feedType": "mrss-json", "title": "Remote DASH Feed", "url":
48         "https://s3.amazonaws.com/feeds.atcsc3-modelmarket.com/run3tv-dash-feed.json"},
49         {"feedType": "mrss-json", "title": "Local Weather Feed", "url":
50         "/london/Station-3/q-bar/dynamic/weather-feed.json"},
51         {"feedType": "esg", "title": "Local programs", "url": "tag:run3tv.org,2023:globalServiceID/7"},
52         {"feedType": "mrss", "title": "External programs", "url":
53         "/london/Station-3/q-bar/dynamic/programs.atom"},
54         {"feedType": "mrss", "title": "VAST", "url":
55         "https://example.com/feeds/df3083f49bc31d12dd12246fb33edf9600426754", "ads": {"vastTag":
56         "https://serveit.linkstorm.net/sb/ETHE_DEMO_RXVP/linear.xml"}},
57         {"feedType": "mrss", "url": "https://example.com/feeds/8e5146c3aeb2532f59cdd64187ec19215b541c38"},
58         {"feedType": "mrss", "url": "https://example.com/feeds/0429228931719e6b195b448fe17b9030f632d3a4"},
59         {"feedType": "mrss", "url": "https://example.com/feeds/047fefe0325dfac431df5731d9001e33824ed9d7"},
60         {"feedType": "mrss", "url": "https://example.com/feeds/bd23363a9ab9a2829e51fdcf82ae484a4142d21a"},
61         {"feedType": "mrss", "url": "/london/Station-3/q-bar/dynamic/settings-<language>.atom"}
62     ],
63     "ticker": {"feedType": "mrss", "title": "TICKER", "speed": 10, "url":
64     "/london/Station-3/q-bar/dynamic/ticker.atom"}
65 }

```

In this example, a Q-Bar application is configured to:

- Timeout at set points (lines 4 - 9)
- Be styled in a specific way (lines 14 - 38)
- Include a number of feeds (lines 42 - 54) of content (including a “feed” linking to the settings section)
- Include a ticker at the bottom of the screen (line 55)

Note The featureset of the Q-Bar application is completely driven by the config.json. For example, removing the `ticker` field will remove the ticker from the application.

9.1. config.json structure

Each Framework application has a standard configuration structure, as described in this section.

For application-specific configuration information, please refer to the documentation for the application.