Run3TV Implementation Guidelines: Common Settings

Version: 2.1
Date: December 18th, 2024
Doc: R3TV-IG-0205

Framework version: 2.4

Revision history

Version	Date	Framework Version	Update
1.0 (Released)	February 5th 2024	2.2	First release
2.0 (Final)	May 29th 2024	2.3	New features for v2.3: ● Additional Alerting configuration in **fmw.json**
2.1 (Final)	December 18th 2024	**2.4**	Added clarification around default behavior for optional fields.

Copyright notice

Contents

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
R3TV-IG-0202	Run3TV	Run3TV Implementation Guidelines: Run3TV Overview
R3TV-IG-0203	Run3TV	Run3TV Implementation Guidelines: Run3TV SDK APIs
A344-2023-02	ATSC	ATSC 3.0 Interactive Content, A/344:2023-02, 17 February 2023
R3TV-IG-0204	Run3TV	Run3TV Implementation Guidelines: DASH Event Streams Specification

3. Introduction

This document describes the Run3TV Framework's Framework-wide configuration options. These options are split across a number of json files. The overall structure of these files is described in Framework configuration file structure.

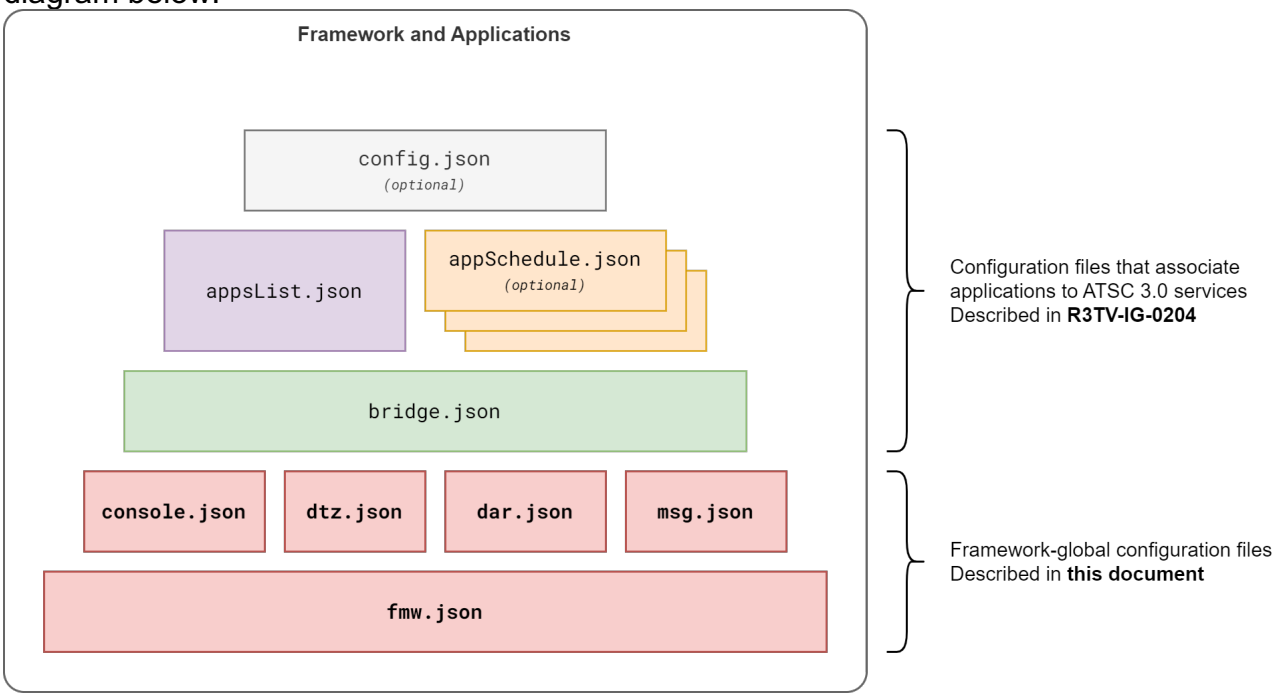
Most configuration fields within these configuration files are mandatory.
Some fields within these configuration files are optional. For these fields, this document describes the "default" configuration that will occur if each field is not present.

> ***Note*** This document can be used in conjunction with R3TV-IG-0204 (available as part of this document pack), which describes service and application level Framework configuration.

It is expected that the reader will have a firm understanding of web development and a reasonable understanding of ATSC 3.0 Interactive Applications.
For more information about the R3TV Framework, please refer to [R3TV-IG-0202].

4. Framework configuration file structure

The Framework makes use of a number of configuration files, as summarized in the diagram below.



This document focuses on the following Framework-global configuration files, all of which can be found in the `public/run3tv-common/conf` directory:

File	Description
`fmw.json`	The core configuration file of the Framework, parsed by the Framework's `index.html`
`dtz.json`	Date and timezone value configuration
`dar.json`	Dynamic Ad Insertion event configuration
`msg.json`	Framework-level error and notification message configuration
`console.json`	On-screen development console configuration

The following sections of this document describe each file in turn.

5. Framework configuration files

5.1. fmw.json (Main Framework configuration)

The **fmw.json** configuration file is the first configuration file to be read when the Framework is first launched. It includes a number of Framework-wide configuration options, as described below.
The Framework's **index.html** file (found at `/public/run3tv-common/index.html` within the Starter Kit) launches and loads fmw.json.
The structure of the file is described in the table below.

Field	Description	Required / Type	Example
`ampPoster`	The image to be displayed when the viewer selects a VOD asset to play. The default provided in the starter kit displays "LOADING" on a black background. This default image can be replaced programmatically using the fmw.amp.setposter() method. For more details see [R3TV-IG-0203].	Base 64-encoded PNG	` iVBORw0KGgoA...`
`vid`	The element ID for the AMP video object within **index.html**	String (element ID)	`vid`
`iot`	*This will be deprecated.* The key sequence used to display the IoT client ID on screen	Array of key identifiers	`["ArrowLeft", "ArrowLeft", "ArrowUp", "ArrowUp"]`

Field	Description	Required / Type	Example
~~`iotOld`~~	~~*Deprecated.*~~ ~~If set to true, the following additional information will be sent to the IOT agent:~~ ~~• Framework state~~ ~~• All JS console output~~ ~~• Current channel information~~ ~~By default this value is true~~	~~Boolean~~	~~`false`~~
~~`debug`~~	~~*Deprecated.*~~ ~~*Previously used to supply a key sequence to display the websocket ready state on screen*~~	~~–~~	
`gtag-debug`	*This will be deprecated.* If set to true, enables real-time Google Tag Manager analytics. ***Note*** Be sure to enable debug mode in Google Tag Manager using the follow command: `dataLayerDebugMode = "debug_mode`	Boolean	`false`
`console`	The configuration object for the on-screen development console	*Console Object*	
`nav`	If true, the navigation buttons will be presented on the on-screen development console. By default this is set to false	Optional Boolean	`true`
`buttons`	A file path to the **console.json** configuration file. This file describes the buttons to present above the on-screen development console. Further details can be found in console.json (development console configuration)	String (Filename)	`conf/console.json`
`log`	If true, the full-screen logs will be presented on the on-screen development console. By default this is set to true	Optional Boolean	`true`
`max`	The number of lines to store in the on-screen development console log	Integer	`1000`
`show`	If set to true, the Framework will present the on-screen development console once tuned to a channel	Boolean	`true`
`url`	*Deprecated*	–	
`app`	The ATSC 3.0 application object	ATSC 3.0 Application Object	
`open`	If set to true, the Framework will connect to the ATSC 3.0 Interactive Content Command and Control web socket, as defined in [A344-2023-02]	Boolean	`true`
`wsPath`	The location of the ATSC 3.0 Interactive Content Command and Control web socket, as defined in [A344-2023-02]	String	`atscCmd`

Field	Description	Required / Type	Example
dynamic	The Framework's dynamic application object	Dynamic Application Object	
iframe	The element ID for the iframe that will present the dynamic application, as defined in **index.html**	String (element ID)	`ifr`
delay	The time (in seconds) to pause between stopping one application and starting a new one	Integer	`1`
bridge	The location of the bridge.json file. This file is discussed in detail in [R3TV-IG-0204]	String (Filename)	`/bridge-conf/1.0/bridge.json`
exception	The Exception App object. This object describes the application that will be launched if the **appsList.json** file is corrupt or missing. The fields within this object follow the same structure as those within appsList.json. For more details, please see [R3TV-IG-0204]	Exception App Object	
appRoot	The local directory path to the base of the exception application	String (Path)	`apps`
appEntry	The filename of the exception application's entry HTML file. This value will be appended to the `appRoot` value to create a complete file path	String (Path)	`/run3tv-exception.html/?params=<SOURCE>&callsign=<callsign>`
appVersion	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		`1.0.0`
appId	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		`com.run3tv.exception`
properties	The configuration properties for the exception application. This can be supplied as a file path or as a json object	Optional Properties Object/String	
eventStream	The Framework Event Stream object. Used to subscribe to Framework event streams. For more details, see [R3TV-IG-0204]	Framework Event Stream Object	
schemeIdUri	The schemeIdUri value of the Framework events to listen for	String	`tv:eventstream.run3tv.org`

Field	Description	Required / Type	Example
subscribe	If set to true, the Framework will subscribe to the event stream (using the `schemeIdUri` value above). When events occur, the Framework will notify the application	Boolean	`true`
popUp	The dynamic applications' on-screen notification / message window configuration and styling definition. The styling within this object defines the popups presented to viewers when the `fmw.notify.send()` function is called. For more details, see [R3TV-IG-0203]	PopUp Object	
element	The element ID for the pop-up object, as defined in **index.html**	String (element ID)	`pop`
popStyle	Styling details for the dynamic applications' on-screen notification / message window	PopUp Style Object	
pop	CSS styling for the pop-up window	String (CSS)	
img	*Deprecated*	–	
title	CSS styling for the title of the pop-up window	String (CSS)	
txt	CSS styling for the body text of the pop-up window	String (CSS)	
btn	CSS styling for the wrapper around the pop-up buttons	String (CSS)	
button	CSS styling for the pop-up buttons	String (CSS)	
code	*Deprecated*	String (CSS)	
msg	The file location of the **msg.json** file. This file contains configuration information for notification and error messages. Further details can be found in msg.json (Notification and error messaging configuration)	String (Filename)	`conf/msg.json`
dtz	The file location of the **dtz.json** file This file contains date and time zone configuration information. This data is used by the Framework when reporting time zone information to Google Tag Manager (replacing the ATSC 3.0 / JS timezone type of "GMT+xxxx" with human-readable time zone identifiers) Further details can be found in dtz.json (Date and TimeZone configuration)	String (Filename)	`conf/dtz.json`

Field	Description	Required / Type	Example
dar	The file location of the **dar.json** file This file contains dynamic ad insertion configuration Further details can be found in dar.json (Dynamic Ad configuration)	String (Filename)	`conf/dar.json`
token	The identifier of the broadcaster. This value is generated by Run3TV and supplied to the broadcaster. For more details, please speak with your Run3TV representative	String	
iotHeartbeats	This contains an array of objects containing information how long to send a heartbeat and the interval of the heartbeat. The order matters as the first entry in the array is configured first, once the duration has elapsed, then the next object is used and the heartbeats configured accordingly.	Array of Objects	`[{},{}]`
Duration	The number of seconds for the heartbeat to run before before the next configuration is applied. -1 indicates that this is the last object and continue indefinitely. 0 shall be skipped if applied. This value should be divisible by the interval.	Number (Seconds)	`60`
interval	This value indicates how often to issue a heartbeat to the RUN3TV platform. If the duration has elapsed and the system has been moved to the next configuration, the last heartbeat may not be fired.	Number (Seconds)	`10`
dataAllow	An array of regex values. If the receiver's useragent matches any of these regexes, then IOT reporting data is sent to the IOT agent. This data is a duplicate of the data sent to Google Tag Manager	Array of Regex strings	`["/"]`
alerting		Framework Alerting Object	
enabled	If set to true, ATSC 3.0 AEAT and OSN alerts will be displayed by the Framework as an on-screen ticker. This is presented in the form: `"<AEAText> | <EventDesc>"` ***Note*** The viewer can configure which alert display using the properties dialog. T dialog can be presented using `fmw.input.preferences()`. The application can receive notificati about these alerts by registering for using the `onAlert()` event	Boolean	`True`

Field	Description	Required / Type	Example
style	Styling for the alert message	String (CSS)	
liveTV	Styling to use when presenting alerts over live TV when a Framework application is not running	AlertingStyle Object	
topApp	Styling to use when presenting alerts over Framework applications	AlertingStyle Object	
appLink	The app to launch when the viewer presses **OK** on an alert. If no app is provided, the **OK** button is used to close the alert	Optional String (appName ID)	`AlertApp1`
properties	The configuration properties for the app to launch when the viewer presses **OK** on an alert	Optional Properties Object	
fmwkSnoozeModes	An optional configuration to hide or replace an application based on manufacturer, model or user agent	Optional Array of Framework Snooze Mode Objects	
condition	An object that describes receiver conditions to be met for the application to be swapped out. If all conditions within this object are tested to be true, the application will be swapped out with the application listed in the `appLink` field below At least one of the condition fields below must be included in this object	Object	
device	An optional RegEx that will be matched against the receiver's User Agent string. If the match is not successful, the Framework will skip this `AppSnoozeModes` object	Optional String (Regex)	`Samsung|Tizen`
ccEnabled	An optional check against the current closed caption state of the device. For example, if this field is set to true and the receiver has closed captions enabled, the check will be considered successful. This check is useful to use if receivers are only able to provide a good viewer experience when closed captioning is disabled	Optional Boolean	`true`
appLink	The appName of the replacement application to launch, as defined in **appsList.json**	String (appName ID)	`ExceptionApp`
properties	The configuration properties for the replacement application	Optional Properties Object	
privacy	An object that defines the location of the various legal texts. For more details on this and the Replaceable Field Tags that can be used to dynamically populate information within these files,, refer to [R3TV-IG-0204]	Privacy Object	
exception-legals		Object	

Field	Description	Required / Type	Example
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 LLC`
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`
settings	This array provides the DEFAULT privacy rules if the framework has been launched on a service that has not been explicitly configured for a specific market. Note: Typically these settings are taken from the AppList.json of the market that has been configured for the specific service rendering the application.	Array	
id	Unique identifier for the object to be displayed in the common settings. Three values are required. • default_terms • default_privacy • default_optin	String	`default_terms`
type	The type of window to display in the settings panel. Two types are supported. • html (multiple entries) • optin	String	
menuTitle	The menu label to display on the settings panel. This is a multi-lingual object following the standard convention.	Object	`{"en": "Terms of Use", "es": "Condiciones"}`
pageTitle	This attribute configs the title on the settings panel. This is a multi-lingual object following the standard convention.	Object	`{"en": "Terms of Use", "es": "Condiciones de uso"}`

Field	Description	Required / Type	Example
url	Required if type = html The URL to fetch html page to render in the panel. The URLwill support TAG REPLACEMENT.	String (URL)	`https://public.a3fa.yottacloud.tv/run3tv/pp-<language>.html`
inaccessible	Required if type = html This is show if the document can not be accessible by the user. This is a multi-lingual object following the standard convention.	Object	`{"en":"Document inaccessible.", "es": "Documento inaccesible."}`
para	Required if type = optin This text is display in the optin panel to provide information to the viewer. This is a multi-lingual object following the standard convention.	Object	`{"en":"paragraph of text", "es": " paragraph of text"}`
opts	Required if type = optin This object contains an array of opt-in options that the panel should display. The settings panel are only aware of the id's provided in the row below.	Array	
id	Required if type = optin There are three unique identifiers required • DataSharingConsent • UserProfilingConsent • AdvertisingConsent	String	`DataSharingConsent`
label	Required if type = optin This attribute provide the messaging to describe the option that the viewer can change. This is a multi-lingual object following the standard convention.	String	`{"en": "Do Not Sell or Share My Personal Information", "es": "No venda ni comparta mi información personal"}`
defaultOptIn	Required if type = optin When the framework first time runs, the privacy must have a default value based on the broadcaster's legal requirements.	Boolean	`false`
logging	An object that defines the logging settings for the Framework	Logging Object	

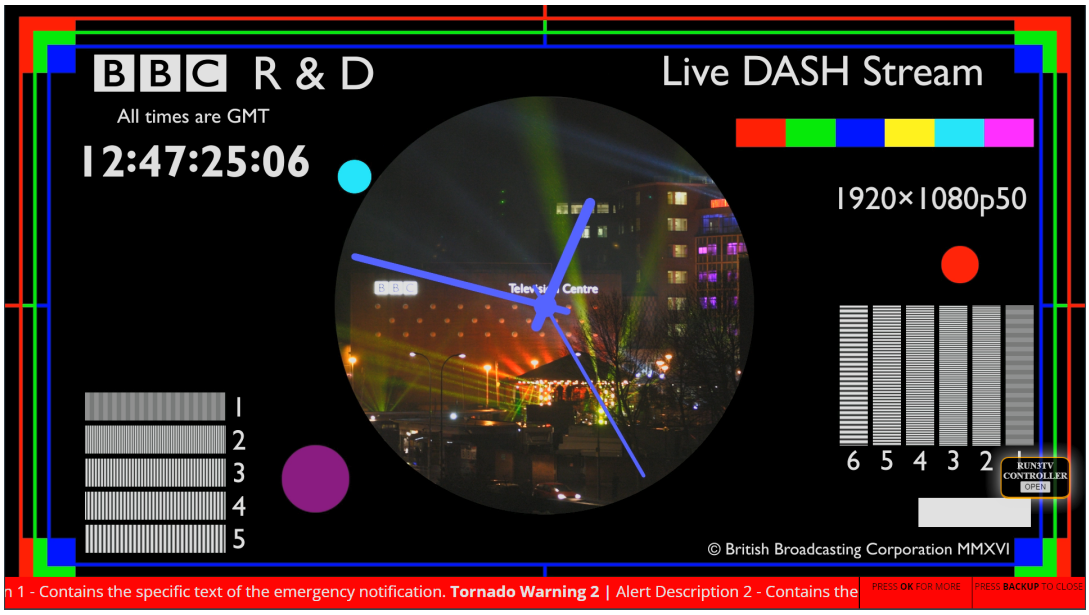
Field	Description	Required / Type	Example
`exception`	If set to true, only events of type “frwkException” are sent to Google Tag Manager. Otherwise, all events will be sent to Google Tag Manager. Generally, in production this should be set to true	Boolean	`false`

5.1.1. AlertingStyle Object

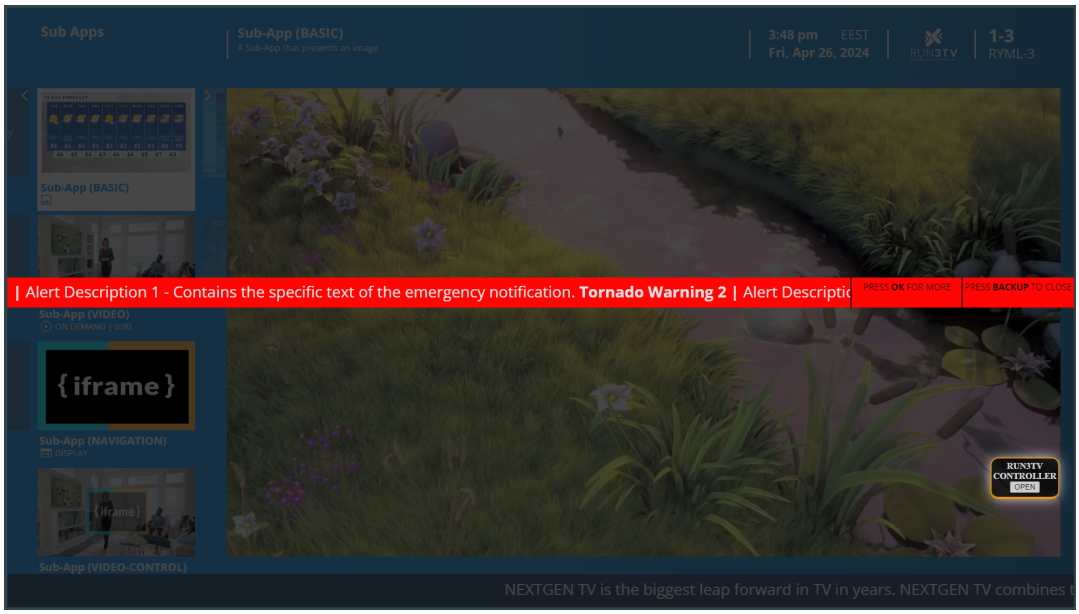
The styling for alerts is defined as follows:

Field	Description	Required / Type	Example
`scrollText`	CSS Styling for the scrolling alert text	String (CSS)	`font-family: 'Open Sans', sans-serif;background-color: red;color: white;position: absolute;display: block;width: 100%;height: 56px;font-size: 28px;bottom: -6px;padding-top: 6px;`
`selectButton`	CSS Styling for the alert's Select button	String (CSS)	`font-family: 'Open Sans', sans-serif;background-color: red;color: black;position: absolute;display: block;width: 200px;height: 50px;font-size: 16px;bottom: 0px;right: 200px;padding-top: 6px;border: none;border-left: 2px solid black;text-align: center;`
`selectLabel`	Text to present on the alert's Select button	String or Multi language string Object	`PRESS <strong><baAppearLabel></strong> FOR MORE`
`backButton`	CSS Styling for the alert's Back button	String (CSS)	`font-family: 'Open Sans', sans-serif;background-color: red;color: black;position: absolute;display: block;width: 200px;height: 50px;font-size: 16px;bottom: 0px;right: 0px;padding-top: 6px;border: none;border-left: 1px solid black;text-align: center;`
`backLabel`	Text to present on the alert's Back button	String or Multi language string Object	`PRESS <strong>BACKUP</strong> TO CLOSE`

An example of an alert presented over live television is shown below:



An example of an alert presented over a Framework Application is shown below:



5.1.2. Framework Snooze Modes

The optional `fmrkSnoozeModes` object enables the replacement of the application based on the type and/or state of the receiver.
This functionality can be used to stop the usual application from launching on a subset of receivers that may be known to provide a poor viewer experience.

Note Fmrk Snooze Modes enable replacement of all applications based on receiver types and states.
It is possible to perform replacement of specific applications based on receiver types and states by using Application Snooze Modes (configured in the **appsList.json** file).
For more details see [R3TV-IG-0204].

Zero, one or more `AppSnoozeMode` objects can be provided.
At the point when the Framework attempts to launch an application on a channel as defined in **appsList.json**, it will check each `AppSnoozeMode` object in turn for that application. If the receiver matches all of the checks within an `AppSnoozeMode` object, the Framework will launch the replacement application listed in that `AppSnoozeMode` object rather than the application defined in the **appsList.json** file.

Note Care should be taken to avoid defining:

- `FmrkSnoozeMode` objects that refer to replacement applications that also make use of additional AppSnoozeMode (see [R3TV-IG-0204]) tests.
- `AppSnoozeMode` objects that refer to replacement applications that also make use of additional `AppSnoozeMode` tests that could result in an infinite loop of application substitutions.

Either configuration could result in an infinite loop of application substitutions.

Example 1 describes how to substitute any application on all Tolka receivers with the *run3tvSnoozeModes* application, but only when closed captioning is enabled.
The Framework will perform this substitute on Tolka receivers in all of the following situations:

- When first intending to launch an application (for example: after a channel tune) whilst closed captioning is enabled; or,
- When *any* application is running, and the viewer has enabled closed captions.

FmrkSnoozeMode Example 1: Substituting applications on Tolka receivers when closed captions are enabled

fmw.json

```
{
  ...
  "appSnoozeModes": [
    {
      "condition": {
        "device": "Tolka",
        "ccEnabled": true
      },
      "appLink": "run3tvSnoozeModes",
      "properties": {
        "messageType": "redlist",
        "tracking": true
      }
    }
  ]
  ...
}
```

Example 2 describes how to substitute any application on all Tolka and Samsung receivers with the *run3tvSnoozeModes* application. This example includes two separate `fmrkSnoozeMode` objects within the `fmrkSnoozeModes` array; one for Samsung receivers (lines 4 - 13) and one for Tolka receivers (lines 14 to 23).

FmwkSnoozeMode Example 2: Substituting applications on Tolka and Samsung receivers

fmw.json

```
1   {
2     ...
3     "appSnoozeModes": [
4       {
5         "condition": {
6           "device": "Samsung|Tizen"
7         },
8         "appLink": "run3tvSnoozeModes",
9         "properties": {
10          "messageType": "redlist",
11          "tracking": true
12        }
13      },
14      {
15        "condition": {
16          "device": "Tolka"
17        },
18        "appLink": "run3tvSnoozeModes",
19        "properties": {
20          "messageType": "redlist",
21          "tracking": true
22        }
23      }
24    ]
25    ...
26  }
```

In this example, when launched, the run3tvSnoozeModes application will report back that it has launched using the `messageType` “redlist”. For more information about the run3tvSnoozeModes application, see [R3TV-IG-0202].

5.2. dtz.json (Date and TimeZone configuration)

The **dtz.json** configuration file provides the Framework with a mapping of time zone offsets to human-readable time zone codes.
When launching in a new market, or in advance of a new time zone being introduced into an existing market, this file must be updated to include all time zone values applicable to receivers in that market.
The structure of the file is described in the table below.

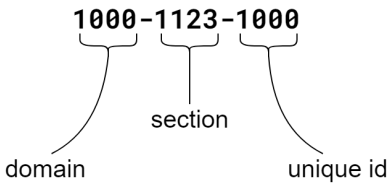
Field	Description	Required / Type	Example
`<GMT Offset>`		Time Zone Object	
value	The human-readable identifier of the time zone, based the tz database For more details, visit: https://www.iana.org/time-zones	String (TZ Identifier)	`America/Anchorage`
code	The colloquial time zone abbreviation. Not currently used. Provided as a comment	String	`EST`

5.3. dar.json (Dynamic Ad configuration)

This section will be included in a future release of this document.

5.4. msg.json (Notification and error messaging configuration)

The **msg.json** configuration file provides fine-grained configuration of Framework-internal notification and error messages.
This file only defines Framework notification and error messages. Application developers should not expand this file for application-specific notifications and errors.
Every message type is defined within msg.json's `list` object using a Unique Message Identifier Triplet taking the numeric form `<domain>-<section>-<unique-id>`.
For example:

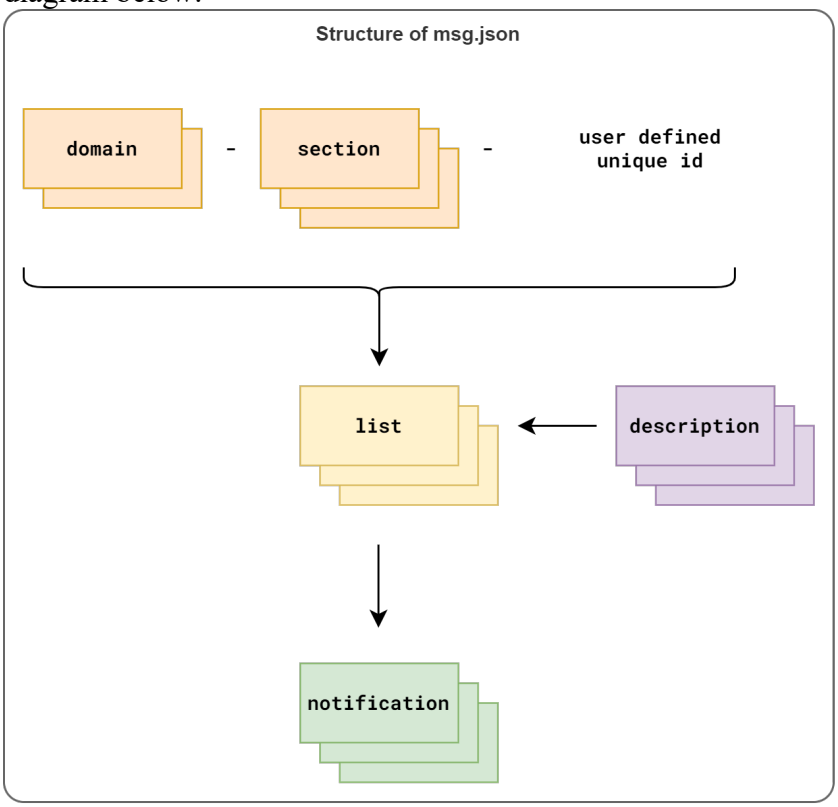


Domains and sections are defined within their respective sections of msg.json. The `unique-id` (the final number in the triplet) needs only be unique for the combination of domain and section. In other words, for differing domains or sections, the same identifier can be reused.
Each message type is given *at least* the following properties:

- A description, containing static (and optionally dynamic) text
- A priority (for more details on priority values, see the notify.send() function in [R3TV-IG-0503])
- A display location, informing the Framework where to present the message
- An error title. By convention, this matches the message's Unique Message Identifier.

As many messages may use the same description text with different data to present, **msg.json** defines a set of standard descriptions within the `description` object that each message can use.

The various levels of indirection within the msg.json configuration file are summarized in the diagram below.



The structure of the file is described in the table below.

Field	Description	Required / Type	Example
error	An object describing the various potential message types within the Framework	ErrorsStructure Object	
debug	Deprecated	–	
default	The default message title text to present	String or Multi language string Object[1]	{"en": "An event occurred during operation", "es": "Se produjo un evento durante la operación"}
support	The default message description to present. If a message does not have a description field, this text will be used	String or Multi language string Object	{"en": "Please visit https://example.com for more information", "es": "Por favor goto https://example.com por ayuda"}
domain	A set of domains	Domain Object	

[1] For more details on Multi Language String Objects, see [R3TV-IG-0204].

Field	Description	Required / Type	Example
<domain IDs : Integer> {1+}	The unique identifier for each organization within the deployment of the Framework. The domain ID forms the first part of the unique message ID triplet	String	`a3fa.fmw`
section	A set of application areas	Section Object	
<section IDs : Integer> {1+}	The unique identifier for each functional area of the application(s). These sections should be chosen to cover areas of the applications at enough of a fine-grained level to be of use when investigating faults	String	`func.localGuide`
description	A set of standard message strings	Description Object	
<description IDs : Integer> {1+}	A string used to describe one or more message types. ***Note*** The association of message to description ID takes place within the `list/description` field Tags of the form `{$__x}` (where x is a digit) can be used to include fault-specific information	String	`fetch: {$__0} with a status code of {$__1}.`
list	A set of standard messages	List Object	
<Unique Message Identifier Triplet> {1+}	An object containing the basic definition of each unique message	Message List Object	
description	The message description ID of the message string to used for the message		
content	An enum that defines which information is presented with the message. The following values are permitted: • `debug` The domain, section and description are presented • `support` Only the `support` string (as defined earlier in this table) is presented • `error-code` Only the error code is presented The default is `error-code`	Enum	`support`
severity	Deprecated	–	
platform	Used as a comment	Optional Enum	`TV`

Field	Description	Required / Type	Example
notification	A set of detailed message configurations that build upon what has been defined within the `list` object	Notification object	
<Unique Message Identifier Triplet> {1+}	An object containing the detailed definition of each unique message	Detailed Message List Object	
priority	The priority of the message. This value is used by the Framework when multiple messages are due to be displayed, to determine which to display first	Integer	`0`
display	The location to display the message. An enum with the following permitted values: • `console` - the development console • `pop-up` - an on-screen pop-up	Enum	`console`
type	The type of the message. An enum with the following permitted values: • error (argument replacement active) • webNotification (argument replacement disabled)	Enum	`error`
error	This value refers the error code as defined within the `error/list` field (above)		
title	The title text of the notification	Optional String or Multi language string Object	`An event occurred during operation 1`
text	Optional body text of the notification	Optional String or Multi language string Object	
image	Optional image	Optional Base 64-encoded PNG	
ttl		Optional Integer	
buttons	An optional array of Button Objects. For more details, see Button Action Object	Optional Array of Buttons Objects	
platform	Used as a comment	Optional	`TV`

5.4.1. Button Object

The Button Object (within the buttons field of msg.json) is used to define the presentation and functionality of the various buttons that can be presented when an error occurs.

The structure of this object is defined below.

Field	Description	Required / Type	Example
<Button Text>	The <Button Text> will be presented within the button. The array of actions will be performed sequentially when the button is activated by the viewer. An empty array indicates that no action should be performed	Array of Button Action Objects	

5.4.2. Button Action Object

The Button Action Object takes the following form:

Field	Description	Required / Type	Example
`action`	An enum describing the action to take. Permitted values are: • `FUNC` JavaScript • `FMW` Calls a Framework-specific function	Enum	`FUNC`
`value`	The command to run	String	`console.log`
`args`	An array of arguments to pass to the command	Array of Strings | Numbers | Booleans	`["OK", "clicked"]`

5.5. console.json (development console configuration)

The **console.json** file provides the Framework with the configuration for the on-screen development console.
The structure of this file is as follows.

Field	Description	Required / Type	Example
`<Anonymous array>`	An array of Console Button objects	Array of Console Button Objects	
`<Console Button>`	An object describing a button that is presented above the development console	Console Button Object	
`<Console button text>`	A JavaScript command to be run when the user selects this button	JavaScript command	`window.fmw.navConsole()`

One or more buttons to be displayed at the top of the development console can be defined in this file.

The JavaScript context for these functions is `top`.

The example file below presents five buttons above the development console, as shown in the screenshot that follows.

console.json example 1

console.json

```
[
    {"consoleClose": "window.fmw.navConsole()"},
    {"clearLog": "window.fmw.clearLog()"},
    {"wsDisconnect": "window.fmw.rpc.disConnect()"},
    {"LIST": "window.fmw.list(console.log)"},
    {"sendLogs(console)": "window.fmw.ajax(window.location.origin +
        '/log-post', window.fmw.conf.console.log.innerText)"}
]
```

Screenshot