



# Creating Shows Programmatically Using Delta Express Mode

User Guide

## Trademark Information

The 7thSense logo, and various hardware and software product names are trademarks of 7thSense Design Ltd. Product or company names that may be mentioned in 7thSense publications are tradenames or trademarks of their respective owners, and such trademarks may also be registered in their respective countries. Windows is a registered trademark of Microsoft Corporation in the United States and other countries.

## Copyright Information

All Rights Reserved. This document is copyrighted © by 7thSense Design Ltd and shall not be reproduced or copied without express written authorisation from 7thSense Design Ltd.

The information in this document is subject to change without notice. 7thSense Design Ltd assumes no responsibility for errors, and/or omissions contained in this information.

## Document Revision

Date	Document edition	Software version	Revision details	Author/Editor
August 2021		Delta 2.7	Initial release	Luke Murray

**M677-1**

[www.7thsense.one](http://www.7thsense.one)  
[info@7thsense.one](mailto:info@7thsense.one)

# Contents

---

Introduction .....	4
Configuring Delta for Express Mode .....	5
The JSON Structure .....	6
Top Level Structure.....	6
Config Structure .....	6
Clips Structure.....	8
Ingesting into Delta .....	11

# Introduction

---

Delta shows can be constructed, managed and run using external control commands. There is an application, StackExpress, that writes these Express Mode shows for you via a simple web browser interface.

However, the JSON files created in this way can also be programmed manually as scripts. These can be loaded into Delta, again using external controls, and with Delta running in Express Mode, the programmed show will run.

# Configuring Delta for Express Mode

---

Express Mode in Delta can be enabled in one of three ways.

## External Control

The quickest method to enable/disable Express Mode is to call the external command:  
**SETEXPRESSMODE ENABLED=[Y/N]**.

## Registry Edit

Alternatively it can be done through the Registry Editor by navigating to:  
**HKEY\_LOCAL\_MACHINE\SOFTWARE\7thSense\Delta**. From there locate the **ExpressMode** and change the value to 1.

## On Load

If neither of these methods is used, Express Mode will be enabled automatically when a media json config file is loaded using the **LOADMEDIAJSON** external control command.

# The JSON Structure

---

## Top Level Structure

```
1  {
2  "tl-1": {
3  >   "Config": { ...
14  },
15  >   "Clips": [ ...
85  ],
86   "timeline": 1
87  },
88  >   "tl-2": { ...
127  },
128   "tlToEdit": 1
129 }
```

At this level, there should only be timeline objects and the `"tlToEdit"` field.

Add a timeline object in the format of `"tl-x": {}` for every timeline you want to create.

The `"tlToEdit"` field specifies which timeline object from the JSON to load into the show.

## Config Structure

```
"Config": {
  "ClipCount": 3,
  "StartFrame": 0,
  "Timeline": 1,
  "KeepDisplayConfig": 0,
  "Layer": 1,
  "Channel": 1,
  "Merge": 0,
  "ClearExpressLayer": 0,
  "Dirty": 0,
  "JSONFileVersion": 1
},
```

### ClipCount

Should match the number of clips in the `"Clips"` array.

### StartFrame

The frame on which to start placing clips.

### Timeline

Should match the number of the outer object, e.g. for `tl-1` this must be 1.

### KeepDisplayConfig

Set to 1 to keep the current display config, set to 0 to overwrite the current display config.

### Layer

The layer on the timeline on which to place the clips.

### Channel

(Not used.)

### Merge

Set to 1 if you have a base file you want to merge the express layer into, rather than completely overwriting the existing file.

### ClearExpressLayer

(If "Merge" is set to 1)

0: Load in clips from this file without clearing the current Express layer.

1: Clear the current express layer before loading in the new clips.

### Dirty

Set to 1 if "Merge" is 1 *and* the clips in this file have changed since Delta last processed this file, otherwise set to 0.

### JSONFileVersion

Should always be 1.

## Clips Structure

```
"Clips": [  
  {  
    "ID": 1000,  
    "Type": "Movie",  
    "MediaType": "tga",  
    "Length": 100,  
    "Name": "7thFlyAnimationHD",  
    "InPoint": 0,  
    "OutPoint": "end",  
    "CrossFade": 0,  
    "PlayMode": "playOnce",  
    "PlayModeDetails": 1,  
    "Size": "FullScreen",  
    "FadeMode": "None",  
    "FadeModeDetails": "both",  
    "FadeModeLength": 30,  
    "PlaySpeed": 1,  
    "DisableCodecAudio": 0,  
    "MatchAudio": 1,  
    "EndTransition": "Play",  
    "StopType": "First",  
    "Actions": [],  
    "LengthMode": "length"  
  },  
]
```

### ID

Used to name the clip on the timeline. Will be formatted as "Clip\_{ID}" : {Name}. Should be sequential. Avoid clashes with other clips.

### Type

The type of resource. Can be "Movie", "Image", or "Capture".

### MediaType

Needs to match the file extension of the media; e.g. "tga", "mp4", "jpg". If "Type" is "Capture" then leave this blank ("").

### Length

The length in frames of the resource. For movie resources, if this does not match the number of frames in the movie, it will be trimmed or looped to meet this value.

### Name

The name of the resource on disk (excluding the extension). If on a movie resource, looks in "C:\Movies", if on an image resource, looks in "C:\Images", if on a capture resource this should be the name of the capture device, e.g. "Integrated Webcam :ID 10000".

### InPoint

Trim a movie resource to start from a specified frame. Must be between 0 and the total number of frames in the movie and always less than Outpoint.



### OutPoint

Trim a movie resource to end on a specified frame. Must be between 0 and the total number of frames in the movie and always greater than InPoint. Set to "end" to not trim the movie.

(Ensure "LengthMode" is set to the correct value when using "Length" vs "InPoint"/"OutPoint".

### CrossFade

Leave as 0.

### PlayMode

"PlayOnce": This clip will play once then move on to the next clip.

"PlayCount": This clip will play the amount of times specified in "PlayModeDetails" before moving on to the next clip.

### PlayModeDetails

The amount of times to loop this clip (only if "PlayMode" is "PlayCount").

### Size

Leave as "FullScreen".

### FadeMode

"None": No transition between clips.

"Transparent": A fade in/out is applied to the clip with a length specified in "FadeModeLength" (seconds).

### FadeModeDetails

Always "both".

### FadeModeLength

Time in seconds to fade in/out of resource (if "FadeMode" is "Transparent").

### PlaySpeed

Sets the play speed of a movie resource. Doesn't affect the length of the resource on the timeline.

### DisableCodecAudio

Only used for Codec Movie resources. If set to 1 then the movie's audio will be disabled.

### MatchAudio

Search for any audio files which contain the same resource name as the movie and add them to the timeline at the same frame as the movie and 4 layers lower.

### EndTransition

"Play" – continue playing after this clip.

"Stop" – stop playback after this clip.

### StopType

"EndTransition" must be "Stop" for this to have any effect.

“Last”: Stop at the end of this clip

“First”: Stop at the beginning of the next clip

### Actions

Place Delta sequences on the timeline

```
"Actions": [  
  {  
    "id": "1",  
    "frame": 20,  
    "tlFrame": 20,  
    "action": "SE_test-sequence",  
    "valid": true  
  }  
],
```

**id** Sequential action ID.

**frame** The RESOURCE frame number to place the sequence on.

**tlFrame** The TIMELINE frame number to place the sequence on.

**action** The name of the sequence to add to the timeline.

**valid** Set to “true”.

### LengthMode

“length” – Use the “Length” field to determine the length of the resource.

“in/out” – Use the “InPoint” and “OutPoint” fields to determine the length of the resource.

# Ingesting into Delta

---

Once the JSON file has been created, copy it into **C:\7thsense\JSON** name it **media.json**.

To trigger Delta to generate the timeline, use the **LOADMEDIAJSON** external control.

This takes one argument, **"tl=x"** where x is the number of the timeline you want to populate. This does not have to match the **"tlToEdit"** field in the top level of the JSON file. **"tlToEdit"** specifies which of the timeline objects to process from the JSON file, whereas the **tl** argument to the **LOADMEDIAJSON** command specifies which timeline within Delta onto which to load content.

If you wish to populate multiple timelines, then it is recommended to store all the timelines in **media.json** (as tl-1, tl-2, etc.) and load them one at a time by updating **"tlToEdit"** to point to the desired timeline, and calling **LOADMEDIAJSON TL=X** to upload it to the correct timeline in Delta.



E: [info@7thsense.one](mailto:info@7thsense.one)

W: [7thsense.one](http://7thsense.one)

**7thSense Design Ltd**

2 The Courtyard Shoreham Road  
Upper Beeding  
Steyning  
West Sussex  
BN44 3TN  
UK

T: +44 (0) 1903 812299

**7thSense LLC, Michigan**

332 E Lincoln Ave  
Suite 100  
Royal Oak, MI 48067  
USA

T: +1 248 599 2717

**7thSense LLC, Orlando**

4207 Vineland Rd  
Suite M1  
Orlando, FL 32811  
USA

T: +1 407 505 5200