Brainsonic Live is relying on an in-house application called Event Engine. For each digital event, an Event Engine instance is created.

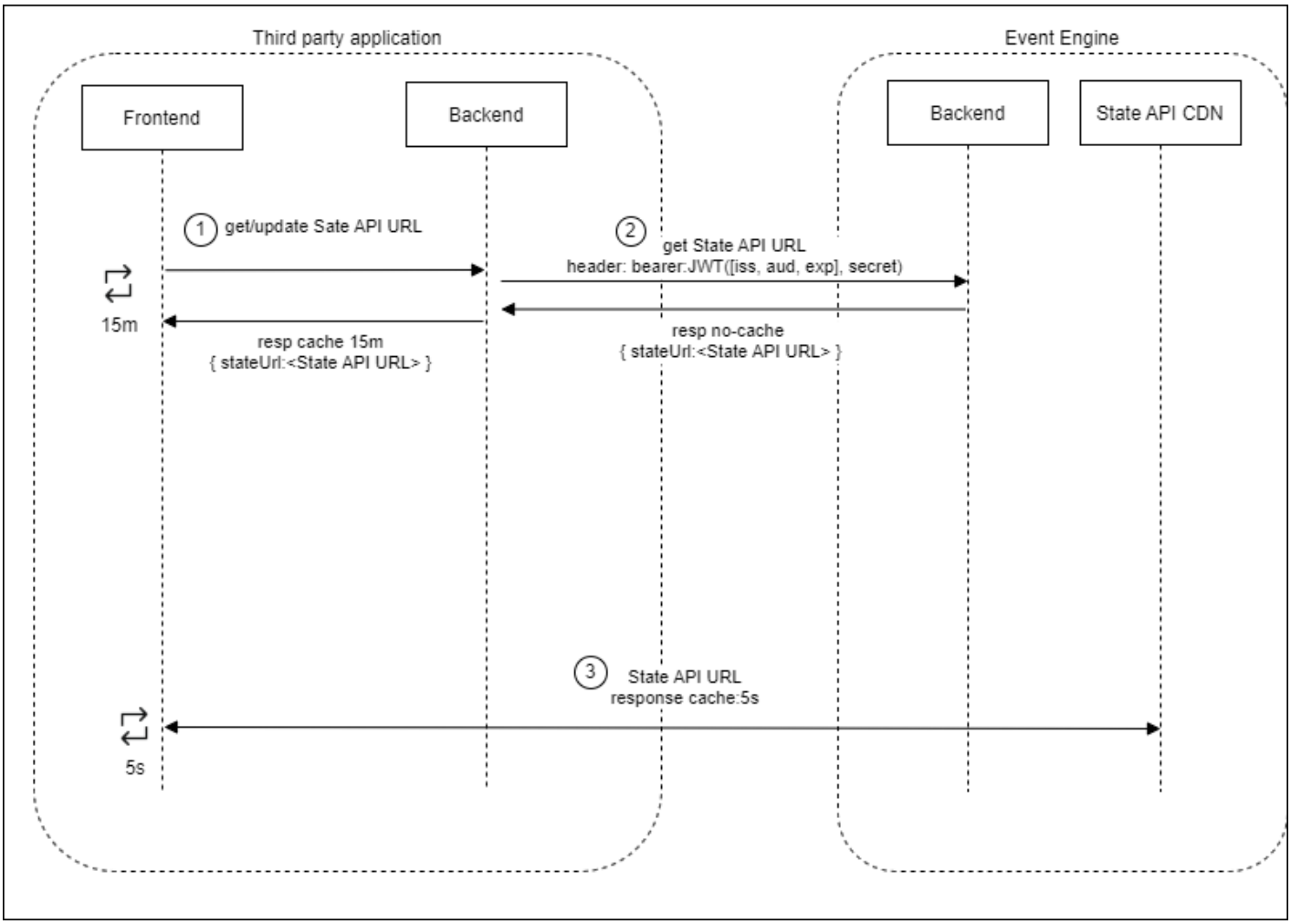
This API allows third party applications to get information about an Event Engine instance configuration and state.

> The access to this API may incur additional charges
> To obtain an API key and secret get in touch with your Brainsonic Live representative

We assume that this third party application is split into a frontend application that is executed on client environment and a backend application that runs on a server.

This information is provided thanks to two requests :

- `getStateURL` intends to be requested from a backend application. This returns the endpoint URL that frontend application will need to request.
- `getState` allows the frontend application to request Event Engine instance state every 5s.



1. The frontend application get the State URL from the backend application
2. The backend application get the Event Engine instance state URL
   - Carries a bearer token. JWT HS512 format,
   - The token is signed with a secret (API key)
   - Payload contains a service ID (`aud`), a client ID (`iss`) and an expiration date (`exp`)
   - TTL range = 30s - 60s
   - To be cached on backend for 15 minutes
3. Every 5 seconds, the frontend application pull the Event Engine instance state
   - Response is cached for 5 seconds

Exemple use case

1. Frontend application request the State URL to backend application (request 1)
2. If the backend application obtained the State URL more than 15 minutes ago then it needs to be requested again (request 2).
   There is no need to maintain a state on the backend server. A 15 minutes cache on request 1 is enough.
3. Frontend application uses the State URL to pull Event Engine instance state every 5 seconds (request 3)
4. Frontend application requests a new State URL to the backend application every 15 minutes.

get State URL

```
requete Method GET
requete Header Authorization: Bearer <JWT>
reponse Cache: no-cache
```

Success response

a json object holding the following properties

- stateUrl `string` URL that need to be requested by frontend application

response Statuses

- Success 200
- Unauthorized Status 401
- Error Status 400

Properties

The token carried by the request is a Json Web Token (JWT) encrypted with a shared secret. Secret is shared between Event Engine and the third party backend application

- alg: HS512
- iss: <your site/application ID>
- aud: <service ID>
- exp: 30s or 60s

get Sate

```
requete Method GET
response Cache 5s
response Status
```

Success response

a json object holding the following properties

- playerState `string` : `custom` | `prelive` | `live` | `postlive` | `replay` | `interruption`
  playerState values relates to the corresponding site states
  - custom: the site displays a custom layout. A typical setup is a countdown as long as a video loop background
  - prelive: on the site a "Live is about to start" waiting card or equivalent is visible
  - live: the live player is loaded and visible
  - postlive: the site displays a "Live has ended" message or equivalent
  - replay: the replay is available
  - interruption: the site displays an interruption "Live will resume soon" message or equivalent

- playerManifest `string` HLS URL
  The manifest loaded by our live player. NB. the value can change from one request to the other.

- countdown `object` Countdown properties
  - visible `boolean` : `true` | `false`
    When true, the countown is visible on our site above the player or waiting card

  - end `unix timestamp` : countdown end time
    When reached the countown displays an end message

  - serverTime `unix timestamp` server current time
    You can adjust end on the client with respect to the local and server time difference

- Example of assets properties
  - customText `Array` an array of strings. The `custom` countdown text split in lines.
  - customTextIsDark `boolean` true when the text color is dark, false otherwise.
  - logoImg `string` URL of the logo image for `custom` states.
  - customUseVideo `boolean` when `true`, hints that the customBgVideo916 is available
  - customBgVideo916 `string` URL of the video to playback (loop) on `custom` state, 9:16 ratio
  - customBgImg916 `string` URL of the background image on `custom` state, 9:16 ratio
  - customFgImg916 `string` URL of the foreground image on `custom` state, 9:16 ratio
  - preliveUseVideo `boolean` when `true`, hints that the preliveBgVideo916 is available
  - preliveBgVideo916 `string` URL of a video on `prelive` state, 9:16 ratio
  - preliveBgImg916 `string` URL of the background image on `prelive` state, 9:16 ratio
  - preliveFgImg916 `string` URL of the foreground image on `prelive` state, 9:16 ratio
  - postliveUseVideo `boolean` when `true`, hints that the postliveBgVideo916 is available
  - postliveBgVideo916 `string` URL of a video on `postlive` state, 9:16 ratio
  - postliveBgImg916 `string` URL of the background image on `postlive` state, 9:16 ratio
  - postliveFgImg916 `string` URL of the foreground image on `postlive` state, 9:16 ratio
  - interruptionUseVideo `boolean` when `true`, hints that the interruptionBgVideo916 is available
  - interruptionBgVideo916 `string` URL of a video on `interruption` state, 9:16 ratio
  - interruptionBgImg916 `string` URL of the background image on `interruption` state, 9:16 ratio
  - interruptionFgImg916 `string` URL of the foreground image on `interruption` state, 9:16 ratio

Response statuses

- Success 200
- Unauthorized Status 401
- Error Status 400

Limitations

get State URL

- Hit/s = 1
- Maximum Hit/minute = 4