MyChannels request forwarding

charif@mychannels.com
daniel@mychannels.com

```
Huidige protocol versie 2.0
Laatste update: 09 July 2018
```

Inleiding

Om MyChannels pagina's integraal in de eigen website te kunnen tonen zullen webservers van de integrator alle requests binnen een bepaald pad (path) (bijvoorbeeld /video of /games) gaan forwarden naar de onze stitchable servers. Vervolgens zullen zij de response verwerken. In dit document wordt de request/response flow beschreven.

Enkele belangrijke zaken:

1. Alle requests gaan over https, het is dus zaak dat de site van de integrator, via https beschikbaar is
2. Alle requests die naar de een stitchable server gaan dienen te worden voorzien van een auth token dat met elke requests wordt meegestuurd.
3. Het principe van *backward compatible and future extensible* wordt gehanteerd, wat inhoudt dat de aanwezigheid en betekenis van de response velden niet zullen wijzigen. Wel bestaat de mogelijkheid om velden toe te voegen. De integrator moet velden waarvan de betekenis nog onbekend is negeren.

Voor de meest basale functionaliteit dient de integrator versie 1 van het protocol te ondersteunen. Voor extended functionality (gebruikers interactie, zoals: bookmarking, volgen, login) dient protocol versie 2 te worden ondersteund.

Definities

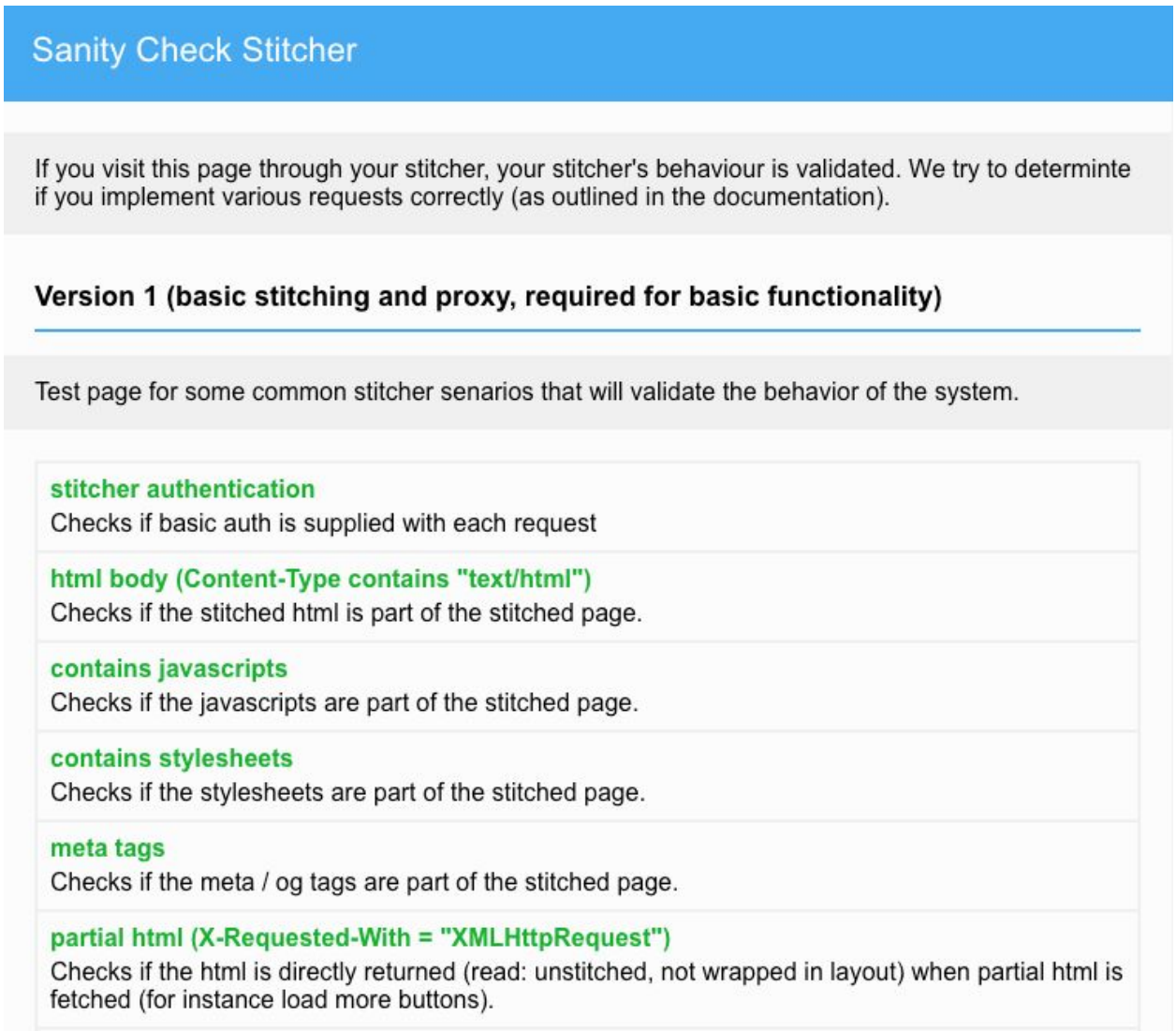
- MyChannels VAAS: Video as a Service platform van MyChannels
- MyChannels GAAS: Games as a Service platform van MyChannels
- MyChannels Stitchable Servers: VAAS of GAAS
- Integrator: De ontvangende partij die MC content gaat integreren in haar site
- Stitching: Dynamische server side include waarbij onderdelen van een JSON response worden gerenderd binnen de layout van de website van de integrator
- Stitcher: Software (draait bij de integrator) welke de responses vanuit MyChannels verwerkt tot een juiste response voor de eindgebruiker (browser of andere client)
- Mountpoint: Het pad (path) waaronder alle stitchable content kan worden opgehaald.
    - Voor de VAAS is dit standaard `/video`.
    - Voor de GAAS is dit standaard `/games`.

Testen van de integratie

Om snel te kunnen checken of de integrator de request / response cyclus op de juiste manier afhandelt, kan het gedrag van de stitcher geverifieerd worden op een URL. Deze draait onder het mountpoint van de integrator op /sanity. Bijvoorbeeld:

https://mychannels.com/video/sanity

mychannels.com is zelf ook een afnemer van de VAAS en heeft derhalve zelf een stitcher draaien. De source code hiervan is opvraagbaar door integrators.



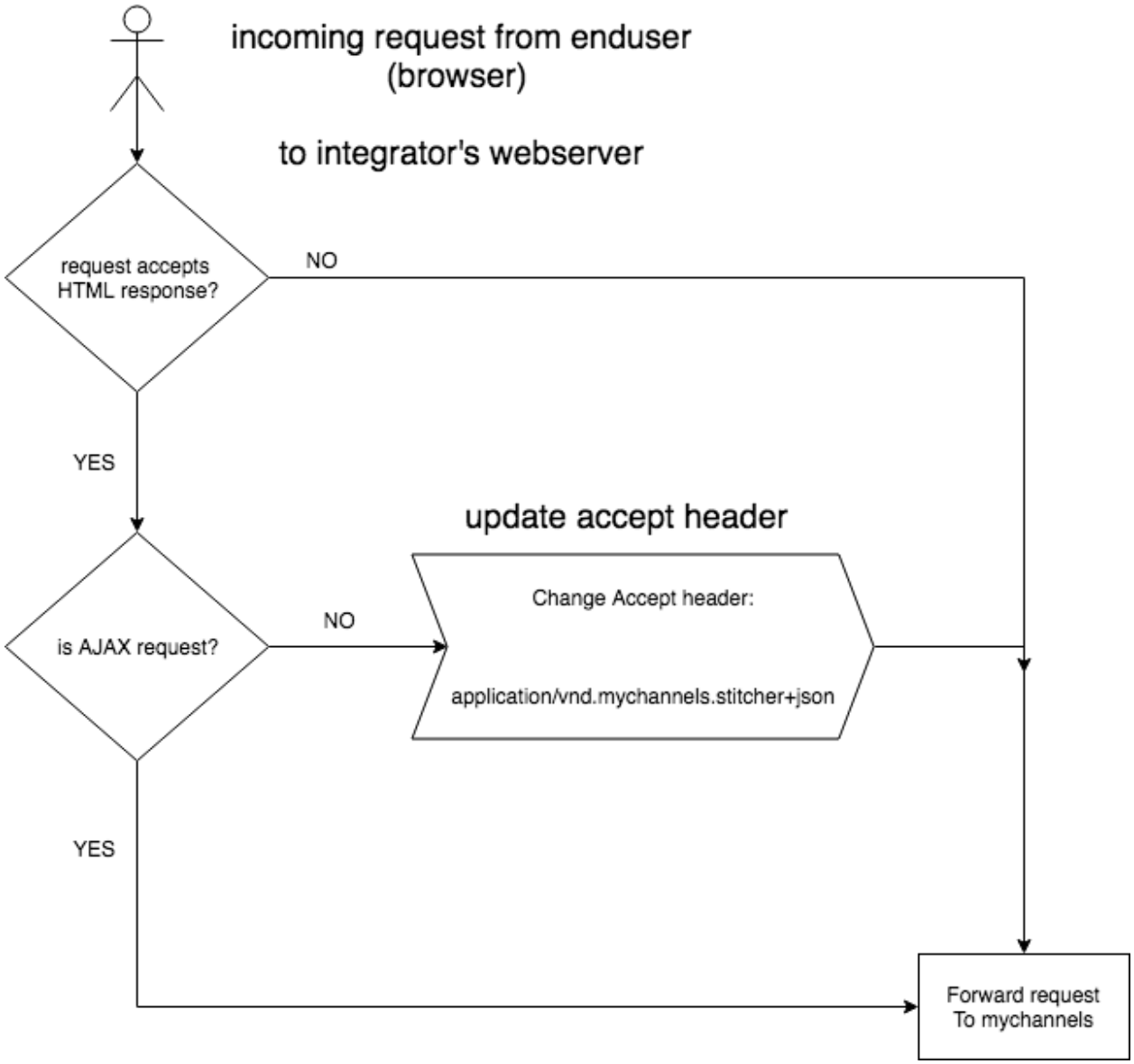
Hier kan de integrator bijvoorbeeld controleren of de page title uit de stitchable servers (Vaas / Gaas) ook echt in de gestitchte HTML terecht komt. En of de styles en JavaScript wel worden

ingeladen. Die de URL voor meer checks. Hieronder volgt een overzicht van hoe de stitcher zich dient te gedragen in verschillende situaties.
De stitcher/proxy van MyChannels is geschreven in NodeJS en ondersteunt reeds protocol versie 2.

Basic Request / Response

1. Een request komt binnen bij de integrator's webserver. Ergens onder het mountpoint, meestal /video (VAAS) of /games (GAAS). Bijvoorbeeld hln.be/video, ad.nl/video of hln.be/games

2. De integrator bepaalt nu zelf of zij de VAAS om een stitchable response gaat vragen of niet. Dit doet zij aan de hand van de volgende flow:



Afbeelding 1: flow diagram wel/geen stitchable response

NB: Als de accept header */* accepteert men feitelijk ook text/html

3. Wanneer geen stitching vereist is kan de request vrijwel *as is* geforward worden. De Host header zal wel moeten worden aangepast naar het domain van de stitchble server (VAAS / GAAS). Bovendien zal er een token moeten worden meegestuurd.
    a. Host header aanpassen:
    Host: yourdomain.mychannels.video
    /
    Host: yourdomain.mychannels.games
    b. Token invoegen
    Authorization: Token 123

4. Wanneer een stitchable response vereist is dient de request iets aangepast te worden:
    a. Host header aanpassen:
    Host: yourdomain.mychannels.video
    /
    Host: yourdomain.mychannels.games
    b. Token invoegen
    Authorization: Token 123
    c. Accept header aanpassen:
    Accept:

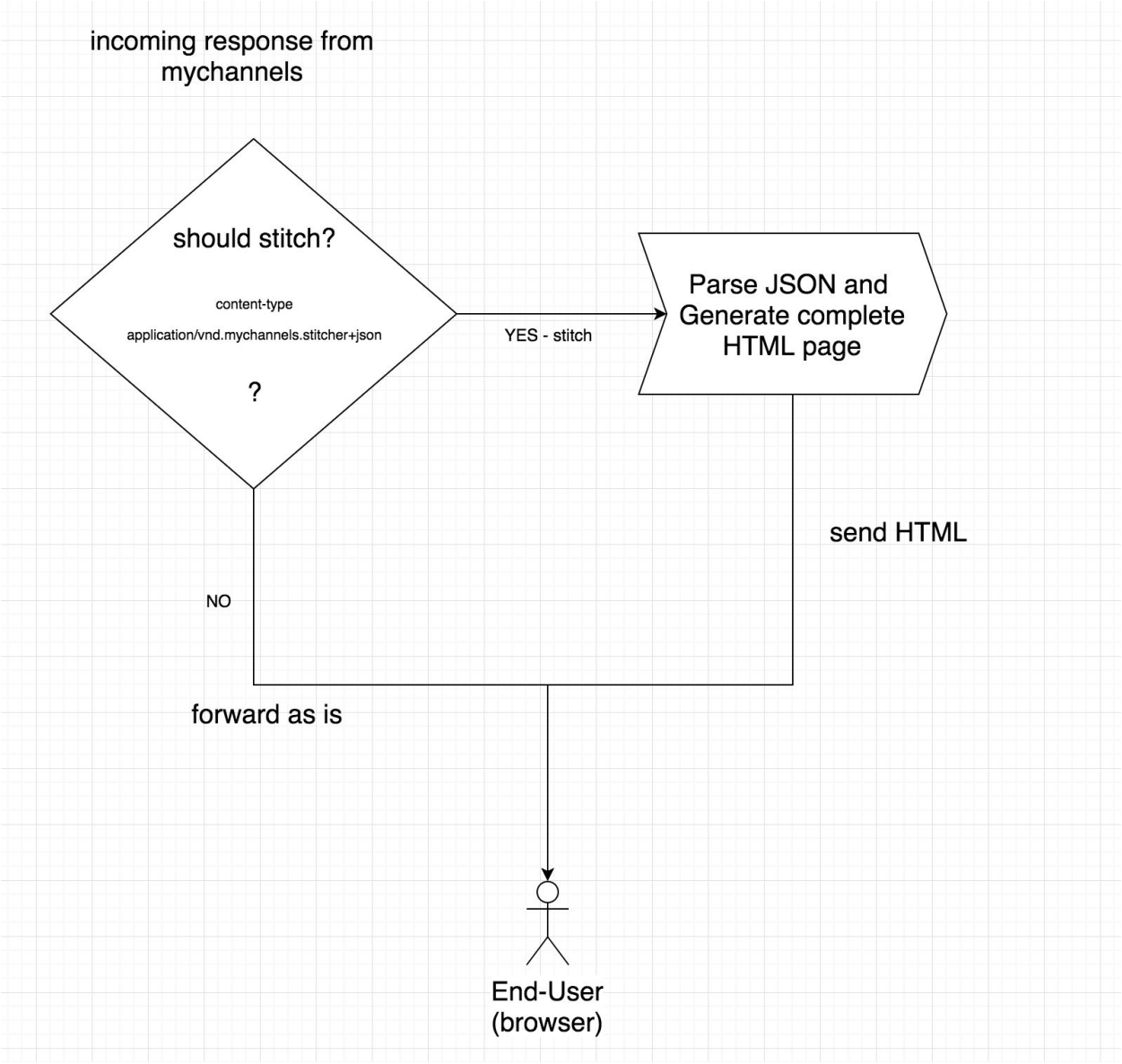
    Dit geeft aan dat de VAAS JSON moet terug sturen, welke door de stitcher kan worden verwerkt tot een volledige HTML pagina met geïntegreerde header en footer.

Afhandeling door de integrator per HTTP response code

Response codes die niet in de 200 range liggen vereisen wat speciale aandacht:

1. Check HTTP response code:
    a. 200 -> Verwerk de response, of stuur deze meteen door (zie flow hieronder) met statuscode 200
    b. 301 -> stuur 301 + Location header door (redirect niet volgen!)
    c. 302 -> stuur 302 + Location header door (redirect niet volgen!)
    d. 404 -> 404 response met eigen 404 pagina
    e. 50x -> 500 / 50x response met eigen 500 pagina
    f. Overige: de oorspronkelijke response code dient te worden behouden

Wanneer de response terugkomt vanuit MyChannels dient deze uiteindelijk weer bij de eindgebruiker aan te komen. Hiervoor kan de volgende flow worden gebruikt:



Afbeelding: 200 OK responses vanuit de VAAS tot aan de eindgebruiker

Headers om host aan te duiden

Om de juiste urls te kunnen genereren, bijvoorbeeld voor de og:url metatag, moet de stitchable server (VAAS / GAAS) weten welke host hiervoor gebruikt moet worden. Daarnaast is het praktisch als de stitchable server (VAAS /GAAS) informatie over de oorspronkelijke client weet.

Dit wordt verkregen via de Forwarded header.

https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Forwarded

`Forwarded: by=<identifier>; for=<identifier>; host=<host>; proto=<http|https>`

De stitchable server (VAAS / GAAS) is met name geïnteresseerd in de *host* en de *for* waarde. In principe zal de host waarde worden gebruikt om canonical urls te genereren. Bijvoorbeeld voor de og:url tag (zie hieronder). Wanneer een andere canonical host gewenst is, bijvoorbeeld wanneer urls altijd naar **integrator.nl** moeten wijzen ook al kijken mensen de pagina's op **www.integrator.nl**, dan kan dat middels de optionele X-Canonical-Host header. Hier dient de host te worden opgegeven, zonder protocol; De stitchable server (VAAS / GAAS) zal altijd https gebruiken.

Voorbeeld request VAAS:

```
curl 'https://hln.mychannels-preview.video/video' \
-H 'accept: application/vnd.mychannels.stitcher+json' \
-H 'authorization: Token somesecret123' \
-H 'host: hln.mychannels-preview.video' \
-H 'forwarded: host=www.example.com; proto=https' \
-H 'x-canonical-host: example.com'
```

`application/vnd.mychannels.stitcher+json`

`application/vnd.mychannels.stitcher+json'`

Voorbeeld request GAAS:

```
curl 'https://hln.mychannels-preview.games/games' \
-H 'accept: application/vnd.mychannels.stitcher+json' \
-H 'authorization: Basic
aGxuX2FwaUBteWNoYW5uZWxzLmNvbTpXam1UVGh2NSV9YnN3YHY=' \
-H 'host: hln.mychannels-preview.games' \
-H 'forwarded: host=www.example.com; proto=https' \
-H 'x-canonical-host: example.com'
```

Stitchable response

Wanneer om een stitchable request is gevraagd zal de stitchable server (VAAS / GAAS) antwoorden met een JSON response body, met hierin de volgende velden:

- Een lijst van 0 of meer https urls naar minified CSS bestanden op een CDN
- Een lijst van 0 of meer https urls naar minified JavaScript bestanden op een CDN

De waarde zal altijd een array zijn, maar kan dus een lege array zijn.

- HTML: string van de content welke door de stitcher zal worden opgepikt
- meta tags:
    - Title
    - Properties
    - Names

Alle keys zullen aanwezig zijn, maar de value mag “”, of een lege array [] zijn.

Voorbeeld response:


```
{
  "stylesheets": [

"https://mychannels-preview.video/css/app-92ed962122ae463c6f373b3d32888ec2.css?vsn=d",

"https://mychannels-preview.video/css/themes/hln-35ee1e8c12c34243e16706b4466fa562.css?
vsn=d"
  ],
  "javascripts": [
    "//imasdk.googleapis.com/js/sdkloader/ima3.js",

"https://mychannels-preview.video/js/app-4dfb6b553587c51dcef2ca6e99f63a72.js?vsn=d"
  ],
  "meta": {
    "title": "Viral Videos",
    "properties": [
      {
        "value": "Elke dag de 3 beste viral video's gebundeld bij elkaar.",
        "key": "og:description"
      }, {
        "value":
"https://mychannels.imgix.net/imgix/06738e9f-c23e-46b6-8e6f-90206e7e1e67/sh_Viral3.jpg
?fit=crop&auto=format%2Ccompress&fm=jpg&w=1200&h=630&dpr=1&crop=focalpoint&fp-x=0.5&fp
-y=0.5",
        "key": "og:image"
      }, {
        "value": "630",
        "key": "og:image:height"
      }, {
        "value": "1200",
        "key": "og:image:width"
      }, {
        "value":
"https://hln.be/video/kanalen/het-laatste-nieuws~c295/series/viral3-2~s880",
        "key": "og:url"
      }
    ],
    "names": [
      {
        "value": "Elke dag de 3 beste viral video's gebundeld bij elkaar.",
        "key": "description"
      }, {
        "value": "summary_large_image",
        "key": "twitter:card"
      }, {
        "value": "Elke dag de 3 beste viral video's gebundeld bij elkaar.",
```

```
        "key": "twitter:description"
      }, {
        "value":
"https://mychannels.imgix.net/imgix/06738e9f-c23e-46b6-8e6f-90206e7e1e67/sh_Viral3.jpg
?fit=crop&auto=format%2Ccompress&fm=jpg&w=1200&h=630&dpr=1&crop=focalpoint&fp-x=0.5&fp
-y=0.5",
        "key": "twitter:image:src"
      }, {
        "value":
"https://hln.mychannels-preview.video/video/kanalen/het-laatste-nieuws~c295/series/vir
al3-2~s880",
        "key": "twitter:site"
      }, {
        "value": "Viral Videos",
        "key": "twitter:title"
      }
    ]
  },
  "html": "<div>....</div>"
}
```


De HTML node zal nooit een reference naar het stitchable server (VAAS /GAAS) domain mogen bevatten, maar enkel URLs zonder domain, zoals /video/een-leuke-serie of /video/productions/12238, of URLs naar andere domein zoals URLs voor plaatjes. Dit is toegestaan zolang het maar https betreft.

Namespacing en Scoping

Omdat de HTML, CSS en JavaScript gaat worden vermengd met content van de host site zijn er een aantal richtlijnen die kans op performance problemen en conflicterende CSS en JavaScript verkleinen.

1. Host template

Met ‘template’ wordt hier het HTML raamwerk bedoeld waar de content van /video of /games in wordt verwerkt. Het is belangrijk dat dit template zo min mogelijk JavaScript en CSS en HTML bevat. Alleen datgene dat nodig is om het template (waarschijnlijk een header, wat menu’s en een footer) te laten functioneren.

Voor CSS betekent dit dat alleen enkele globale regels (een styleguide) zou moeten hoeven meegeleverd, en de CSS om de hierboven genoemde onderdelen te stylen. Hetzelfde geldt voor de JavaScript. Het is onzinnig en vergroot de kans op conflicten wanneer allerlei CSS en JavaScript wordt ingeladen voor onderdelen die wel binnen de integrator’s website bestaan, maar nooit binnen het mount path (/video, /games) zullen worden getoond. Denk aan formulieren, nieuwsartikelen, registratie schermen etc.

2. JavaScript en CSS scoping

De HTML content die vanuit MyChannels wordt geleverd zal altijd in een div met vaste ID worden gewrapped: `<div id=”mc-root”>...</div>`.

De CSS zullen wij ook altijd hierbinnen scopen, zodat conflicterende CSS moeilijker voor kan komen en in elk geval makkelijk door MyChannels is op te lossen door simpelweg een meer specifieke selector te schrijven. Hetzelfde geldt voor selectors binnen JavaScript.

Een mogelijk probleem is dat de host site bepaald JavaScript gedrag aan de MyChannels content kan hangen. Als daar bijvoorbeeld iets wordt gedaan als `$(“a[title]”).on(“hover”)` dan ondervindt de MyChannels content daar last van. Zaak is dan ook zeer kritisch te kijken naar punt 1, en dit soort scripts niet in te laden op de video pagina’s.

De JavaScript zal zo min mogelijk global scope nodig hebben. Mocht er om één of andere reden toch iets aan het `window` object gehangen moeten worden, dan zal de VAAS dit zoveel mogelijk binnen een `window.mc` object doen.

Cookies `(protocol versie 2)`

Voor sommige functionaliteit zal het nodig zijn dat cookies correct worden geproxied tussen de VAAS en de integrator. Hiervoor is het nodig dat de integrator de waardes uit de Cookie request header doorstuurt naar de VAAS.

POST, PUT, PATCH, DELETE `(protocol versie 2)`

Voor de meest basale functionaliteit (ophalen en tonen van content) hoeft de integrator alleen GET requests te ondersteunen. Voor gebruiker interactieve functies als bookmarken en volgen van shows, opgeven van voorkeuren etc. zal de integrator ook andere HTTP verbs moeten ondersteunen.

Dit is in principe niet anders dan de hierboven beschreven request/response cycle.

Login `(protocol versie 2)`

Voor functionaliteit als bookmarking en volgen van shows zal een gebruiker ingelogd moeten zijn op de site van de integrator. De stitchable server (VAAS / GAAS) dient een unieke identifier van de gebruiker te ontvangen zodat het hier lokaal data aan kan koppelen. Belangrijk: de authenticatie gebeurt puur en alleen door de integrator, de stitchable server (VAAS / GAAS) heeft hier nimmer iets mee van doen. Echter, de stitchable server (VAAS /GAAS) dient met elke request twee custom headers te ontvangen die de gebruiker identificeren.

Voor de VAAS hoeft dit niet per se de interne ID van de gebruiker te zijn, zolang hij uniek is en niet verandert.

Voor de GAAS zou dit ID gelijk moeten zijn aan het ID dat in de Kubus APP gebruikt wordt. Dit ID wordt uit de Jason Web Token gehaald die uit de Gigya service komt.

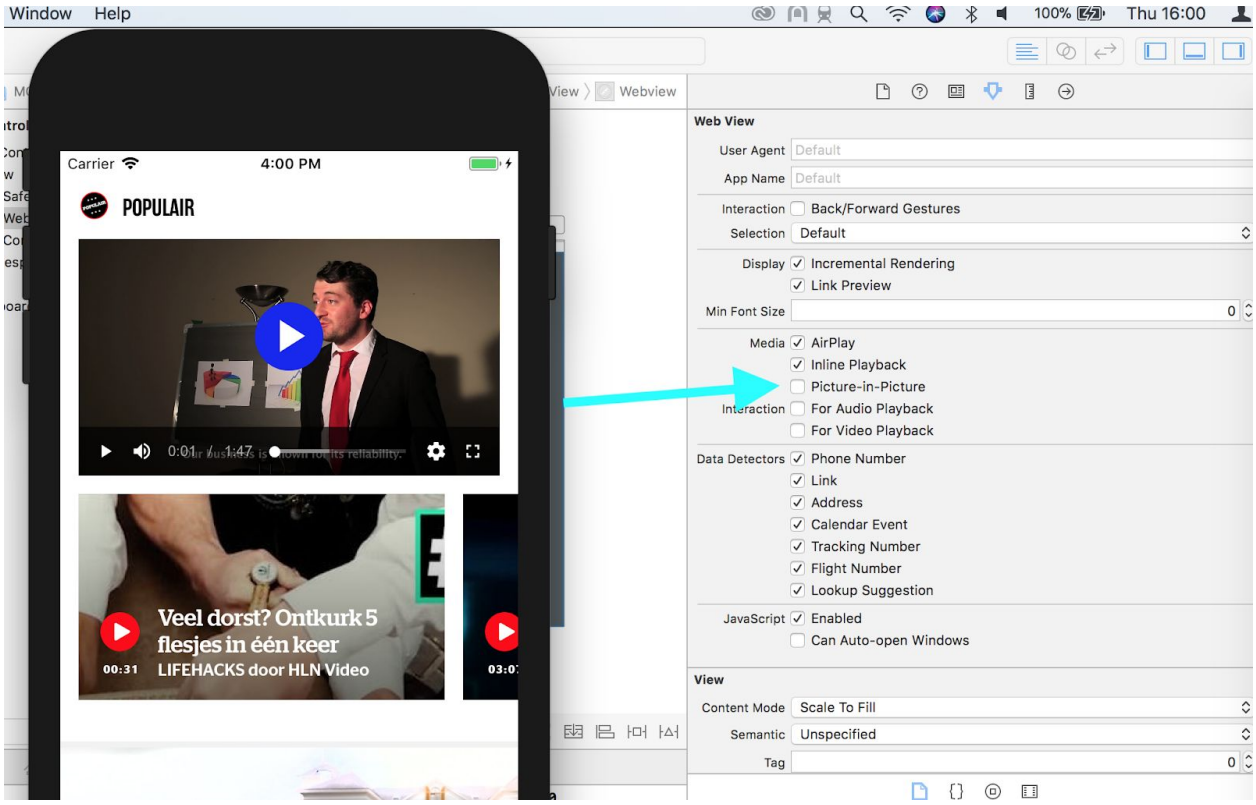
De requests headers zijn de volgende:

Header name	**Summary**	**Example**
External-User-Id	Unieke ID die geshared mag worden met mychannels	`991c0e72-6345-11e8-adc0-fa7ae01bbebc`
External-User-Source	Naam die de bron omschrijft waar de gebruiker uit komt.	`volkskrant-dpg-sso`

Alle HTTP requests header zijn case-insensitive

Webview / Native app integrate

Indien video pages ook in een native webview worden ingeladen is het aan te raden om goed naar de settings te kijken. Niet alle webviews ondersteunen zomaar inline playback van video. De MyChannels videopages zijn wel bedoeld om video inline af te kunnen spelen.



Afbeelding: Voorbeeld van iOS webview in Xcode. Inline Playback is hier ingeschakeld.