Adaptive Portal

Content Delivery Network Proposal

November 2019

Contents

[1 Delivering AP frontend from a CDN 3](#_Toc26779222)

[1.1 Introduction 3](#_Toc26779223)

[1.2 The problem 3](#_Toc26779224)

[1.3 Our proposed solution 3](#_Toc26779225)

[2 Pricing 4](#_Toc26779226)

[2.1 Estimates 4](#_Toc26779227)

[3 Areas of concern 5](#_Toc26779228)

[3.1 It might not actually be possible 5](#_Toc26779229)

[3.2 Hosting AP code outside of the DT network 5](#_Toc26779230)

[3.3 Plugin level custom component development 5](#_Toc26779231)

[3.4 AP deployments not having internet access 5](#_Toc26779232)

[4 Opportunities 6](#_Toc26779233)

[4.1 CDNs are just faster at delivering static files 6](#_Toc26779234)

[4.2 Simplified publishing while still following the new protocols 6](#_Toc26779235)

[4.3 Bleeding edge AP core 6](#_Toc26779236)

[4.4 Switching to project Phoenix 6](#_Toc26779237)

[4.5 No more dotLess 6](#_Toc26779238)

[4.6 Analytics 6](#_Toc26779239)

[4.7 Custom domains 6](#_Toc26779240)

# Delivering AP frontend from a CDN

## Introduction

The frontend dev team would like to move AP Core frontend to a CDN. Using Azure could cost very little but the key benefit would be the saving in dev time, allowing us to deliver frontend fixes much faster and increase performance while reducing the need for so many core nuget package updates in projects (for frontend code anyway).

## The problem

One of the biggest problems in frontend AP dev are nuget package updates.

At the moment if a bug or feature is needed for a project we have to tell that project team it’ll be a week until they can do the nuget update (aligning with the CanadaLife driven weekly core release schedule).

In a project like CanadaLife we are doing weekly nuget updates on a regular schedule, but most frontend tasks are blocked from being marked as resolved while we wait for these to be completed.

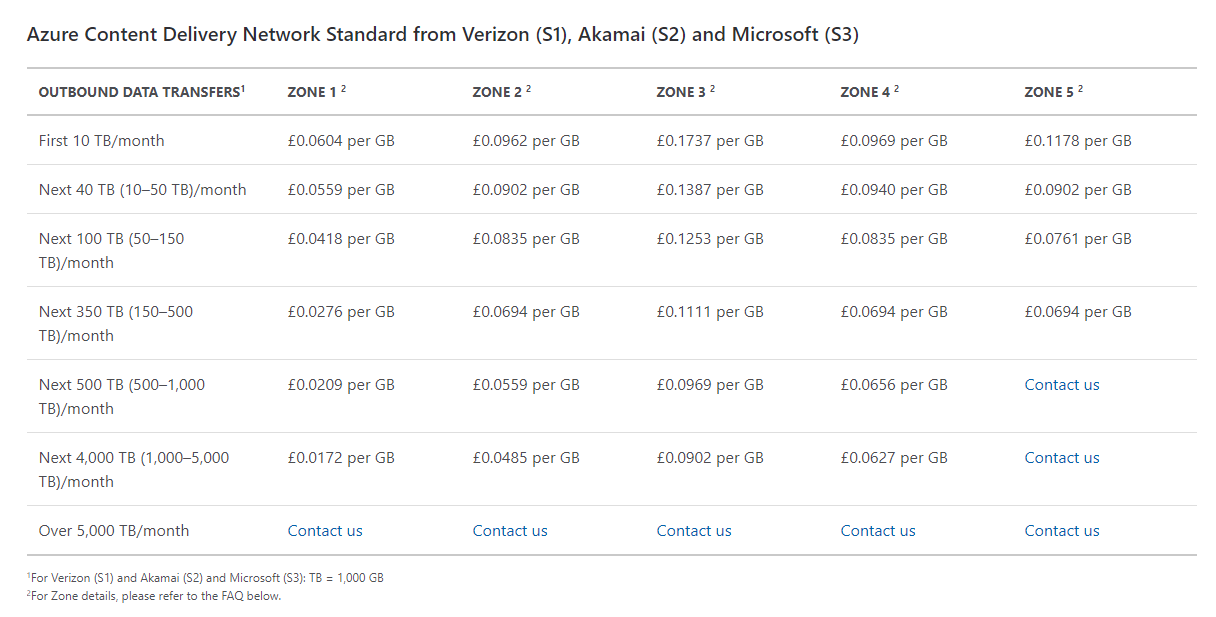
Often that bug or feature is only an hour of actual coding in core, followed by one full dev day to actually get the change into the project that needed it. This proposal will massively reduce that one dev day and keep our developers writing code and not preparing and installing packages.

## Our proposed solution

We were thinking that if we instead hosted the compiled frontend code on a CDN which projects just load via <script> tags everything should work as is, except we could deliver fixes much faster and without time consuming nuget updates and there are some other potential benefits, listed at the end of this document.

# Pricing

The Azure CDN pricing is based on outbound data transfer. In addition to the chart below full details are published online [here](https://azure.microsoft.com/en-gb/pricing/details/cdn/).



## Estimates

Based on the above pricing model (Zone 1, first 10TB) it is estimated that 1 million monthly requests of the Adapt Portal bundle (JavaScript & CSS) will cost **£241.60**.

|  |  |
| --- | --- |
| **Item** | **Value** |
| Price per GB | £0.0604 |
| Price per MB: £0.0604 / 1000 | £0.0000604 |
| AP Core bundle size estimate | 4MB |
| Cost per request: 4 \* 0.0000604 | £0.0002416 |
| Price per million requests: 1,000,000 \* £0.0001208 | **£241.60** |

Please note that for production clients, browser caching will inherently reduce the number of requests made and combined with the fact that AP is a single page application we anticipate that each production client will download the estimated 4mb from the CDN once per release.

We do expect that the development team will make several requests each every day, but even if we have a team of 20 developers each refreshing their browsers (with cache turned off) hitting the CDN every 20mins that equates to less than £1 a month (for the 16,000 requests generated[[1]](#footnote-1)), however we could use our non-production Azure credits so this is really just an example to help understand what a million requests is.

# Areas of concern

## It might not actually be possible

Frontend are planning on using some OKR time to experiment with this idea, and make sure it is feasible.

## Hosting AP code outside of the DT network

1. Can we host our compiled AP framework offsite? (on Azure)

2. Could we also host the non-compiled AP framework offsite?

Another option would be for us to host the files on our own DT servers, might not get the benefits of a full on CDN (unless we can configure a server to be a CDN) but that way we keep all our code in our own domain.

## Plugin level custom component development

We think we would still need to keep frontend in a nuget for projects that are building their own custom components but we would need to update these a lot less, perhaps we could separate frontend source files from backend so core nuget updates for backend fixes don’t also take ages, another potential benefit of this approach.

## AP deployments not having internet access

We could address this in one of two ways, either we could use a custom domain on our CDN or the client could host the files on their own server, either the same one as the backend or on their own CDN.

For either of these to work, we would need to make the base url to the CDN easily configurable to allow us to set this up on a per project basis.

# Opportunities

## CDNs are just faster at delivering static files

Azure CDNs are not expensive so this could cost us very little and would improve performance as delivering frontend files from CDNs would be faster than serving them from the same servers that have our backends on, which under backend load stumble when delivering frontend files.

## Simplified publishing while still following the new protocols

Publishing our code to the CDN would be as simple as FTP-ing the built files to the server, we could make all this happen on the command line and could still follow the new release process in Azure (PRs, numbered release builds etc.) just with an extra step of publishing to the CDN instead or as well as the nuget bit.

This could be a huge time saver for the dev teams too, but there are some business level things that would need approval before we did anything:

## Bleeding edge AP core

Projects under development, like CanadaLife could reference a `latest` version of the core, so would always have the latest frontend code automatically (each time we push to the CDN anyway).

## Switching to project Phoenix

Moving to a CDN will also benefit Project Phoenix as we would host the static frontend files in the same way and could potentially make it simple to switch over to the new ICX frontend framework once it is ready.

## No more dotLess

We currently use the .NET http handler dotLess to compile and serve our styles (written in LESS) however dotLess does have some nuances where it behaves differently from vanilla LESS and its debugging experience is very poor. We would also host the styles (pre-compiled in a build step to CSS) from the CDN.

## Analytics

Azure has some nice analytics around the CDN so we would have good usage stats and it might even be useful to identify bugs or deployment issues.

## Custom domains

Azure CDN fully supports HTTPS and custom domains. E.g.:

<https://ap-core.dunstanthomas.co.uk/latest.js>

1. 8 x 5 x 20 x 20 x 0.0000604 = 0.96p [↑](#footnote-ref-1)