About docs.infor.com content delivery network: CloudFront and content caching

We made some architecture changes with the docs update v 2.9.0 on Nov 23, 2020 that changed the cache behavior of the site. Depending on the last time you viewed the content, it may be cached in an "edge location" – this cache helps with content viewing performance for our customers, but it can be a pain when you are validating your new content.

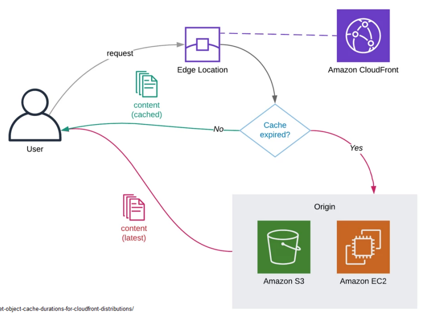
We've adjusted some of the settings in docs Stable deployment, so you should have no problem verifying the content in Stable is new... but when you upload your content to production, if content is cached locally, you won't see the new content immediately, and you need to wait up to one hour after the content was cached in your edge location. One common trick here is to ask a colleague in a different region to check your new content; however, you must remember that if it is also cached in their edge location in the hour preceding your update, they will also see the old, cached content until that cache clears.

Our guideline is: it doesn't hurt to test right away, but if things look weird in production (especially if everything works fine in stable.docs.dev.cloudsuite.com), give it an hour. If your content looks good on Stable, and you see your newly uploaded deliverable in the admin console of the production site (docs.infor.com/adminconsole), it is almost certainly uploaded successfully.   Note: there is no CloudFront caching on the admin console in any of our deployments.

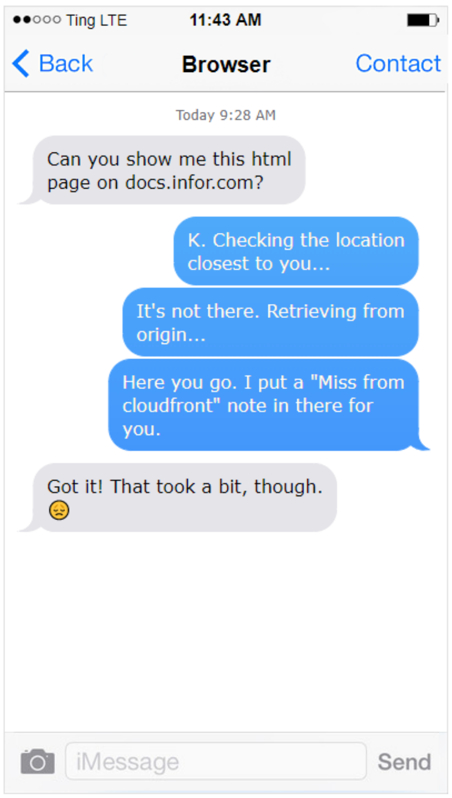
CDN and CloudFront

A content delivery network (CDN) is a geographically distributed group of servers which work together to provide fast delivery of Internet content. A CDN allows for the quick transfer of assets needed for loading Internet content including HTML pages, javascript files, stylesheets, and images.

AWS CloudFront is used as a content delivery service to deliver content to the users around the globe via edge locations. AWS maintains a network of data servers around the world that cache content locally, so that end users are able to access that information more quickly than if they have to go around the world for the origin wherever it may be.



https://blog.shikisoft.com/set-object-cache-durations-for-cloudfront-distributions/



Today 9:28 AM

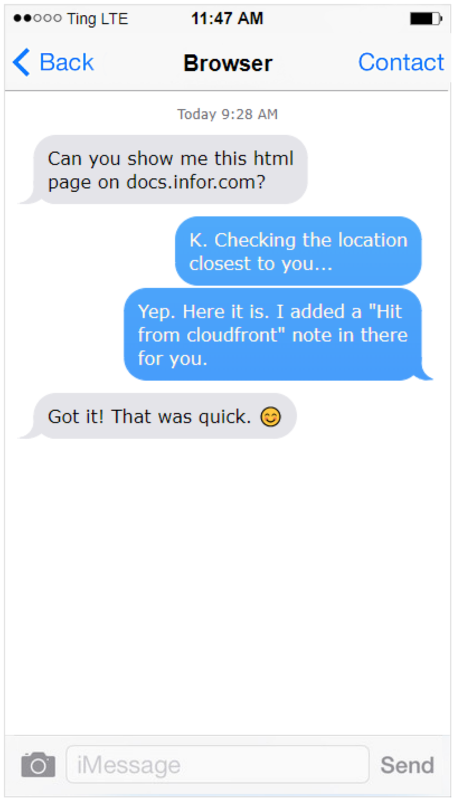
Can you show me this html page on docs.infor.com?

K. Checking the location closest to you...

It's not there. Retrieving from origin...

Here you go. I put a "Miss from cloudfront" note in there for you.

Got it! That took a bit, though. 😣



●●○○○ Ting LTE 11:47 AM

Back **Browser** Contact

Today 9:28 AM

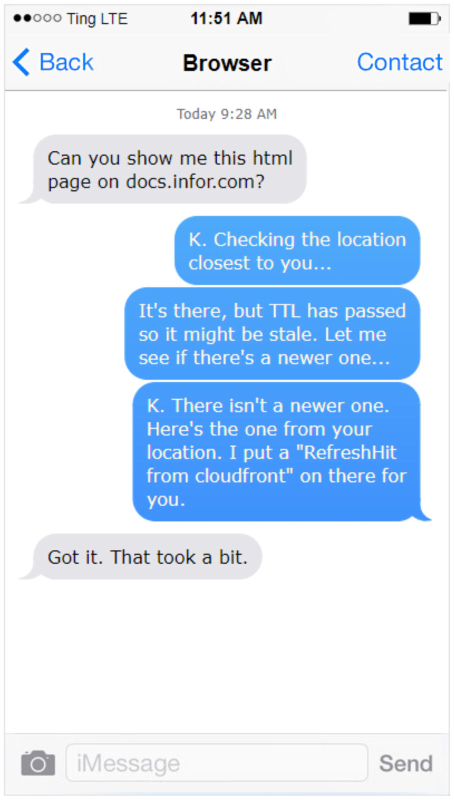
Can you show me this html page on docs.infor.com?

K. Checking the location closest to you…

Yep. Here it is. I added a "Hit from cloudfront" note in there for you.

Got it! That was quick. 😊

iMessage Send



●●○○○ Ting LTE 11:51 AM

Back **Browser** Contact

Today 9:28 AM

Can you show me this html page on docs.infor.com?

K. Checking the location closest to you…

It's there, but TTL has passed so it might be stale. Let me see if there's a newer one…

K. There isn't a newer one. Here's the one from your location. I put a "RefreshHit from cloudfront" on there for you.

Got it. That took a bit.

iMessage Send

Terms

- **Origin** An origin server is the location of the definitive version of an object. Origin servers could be other Amazon Web Services – an Amazon S3 bucket (see below), an Amazon EC2 instance, or an Application Load Balancer.
- **S3 Bucket** The AWS object storage that we use to store your uploaded content. Amazon S3 buckets, which are similar to file folders, store objects, which consist of data and its descriptive metadata.
- **Distribution** This is the CloudFront deployment which is used to route requests from the user to the edge location to the origin server.
- **Edge Location** These are servers located across the globe to reduce the time to deliver content to the end users. These caches can be used to hold the recently requested data. If the data is not available at the edge location, it will be requested from the origin server.

- **TTL** Amazon CloudFront lets you configure a Minimum time-to-live (Min TTL), a Maximum TTL (Max TTL) and a Default TTL to specify how long CloudFront caches your objects in edge locations.

A content delivery network is recommended to:

- improve performance of static content
- provide security measures (such as protection from DDoS)
- enable us to serve secured content by intercepting a request to see if the content is secured – in CloudFront we can use serverless functions called Lambda that run on Edge locations
- provide detailed logs and analytic information about site traffic and visitors – users location, request URIs, referrers, etc (all docs.infor.com traffic dashboard information is based on CloudFront logs)

Read more:

https://aws.amazon.com/blogs/networking-and-content-delivery/improve-your-website-performance-with-amazon-cloudfront/

Cloudfront Implementations: Lessons Learned

https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/HowCloudFrontWorks.html

https://docs.aws.amazon.com/AmazonS3/latest/dev/WebsiteHosting.html

https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/controlling-the-cache-key.html

FAQ

Why do we use CloudFront caching?

For Infor documentation, the final output of our work is static content. Caching makes it much quicker for people to view our content, generally.

If your work output were frequent updates to a web application, then in that case, caching would be counter-productive, because your application and data would need to be 100% synchronized, and caching might break the synchronization. That's why Infor Cloud-based products are deployed in multiple regions without CloudFront caching. However, that is not the case for static html-based content, which you update at most once a month. So we have only one production cloud deployment with over 185 edge locations around the world. It is an efficient global website model that fits our content-delivery needs, while we simply cannot afford to support and maintain 4 or 5 different production deployments globally.

What is the TTL for docs.infor.com?

For our development account where we need to iterate and test content, TTL should be set to 0. In other words, content is not cached in edge locations for stable.docs.dev.inforcloudsuite.com.

For the production account, we use the a TTL of one hour. If content is cached in an edge location, it can stay there for up to 60 minutes. After that TTL expires, it is retrieved from the origin again the next time it is accessed by someone. CloudFront may choose to flush content from edge cache sooner than the default TTL, see CloudFront developer guide for more information.

Update: On 20 April 2022, TTL values for production docs was reduced from 8 hours to one hour. This change was made in Docs 2.1.3.1. On 20 January, we had a meeting to discuss TTL values for production docs with several InfoDev stakeholders and leads. At that meeting it was determined that the TTL should be reduced from the default 24 to 8 hours. This change was made in Docs 2.10.0.

Related Jiras: ~~MTDOCS-1015~~ - CloudFront Caching and Origin Request Policies CLOSED , MTDOCS-1201

My responsive webhelp or doc library home page does not display correctly – it's a bunch of weird boxes and no stylesheet, why does this happen and what can I do?

This is a known issue when responsive webhelp changes or you regenerate and repost your content:

default.html is in CloudFront cache
default.html is looking for files like *_YYYYMMDDTTTT.css
They are not in CloudFront cache
They are requested from S3
They are not in S3 either because new ones replaced them
S3 returns 404
404 is served from cache

We recommend:

- always test in incognito browser session
- wait one hour (which is the current TTL) and try again Note: We have seen those 404 sometimes hang out in cache longer than TTL, we don't know why that is. So you might need to wait and retest again later, or...
- enter a new DOCTOOLS ticket if you cannot get your content to display correctly

More background: to overcome the problems we've seen with browsers caching stylesheets and javascripts and to avoid making customers clear browser cache, we starting versioning some files to force the browsers to download (and keep them) as "new" files. However, when you upload your new content to docs.infor.com, the first step of the upload process actually deletes all the old content files – all the html and css and js that made up your webhelp deliverable. So we've solved the issue of browser caching old files for too long, but now the files aren't around long enough if some older file references them.

Website best practice says CSS and JS are immutable and never change, and are cached for a veeeeery long time – like a year. When they do change, you can give them a new name so they just appear as totally different files to the browser. HTML files topics are mutable, so they can be cached for a time but check to see if they've changed from time-to-time. If HTML includes references to new CSS or JS, no prob, cuz those files are there. If HTML still point to older CSS or JS whether from cache or from server – also no probs cuz those old files are there, too.

We should be keeping all versions of css and js files, but we cannot today. Why? Because they are embedded, many times, within each deliverable and we would have to do incredibly complex diffing on huge content packages (doc libraries) in order to identify all files that were added/updated /deleted and manage them separately. And also develop some something that says, oh yeah, we added a *_202202030800.css but also please keep *_202112300550.css – for reasons. For simplicity and upload performance, the upload to docs does either upload new: add content to S3, add meta data to db OR update existing: delete all files and icua content entries, add current content to S3, add meta data to db.

Implementing best practices requires a serious rearchitecting of our deliverables (DOCTOOLS-12238).

We have to approach this with a long term and short term solution: The long term is the separation and management of framework CSS and JS from content-specific files: DOCTOOLS-12238. The short term solution is to shorten CloudFront cache TTL.

Related Jiras: **DOCTOOLS-12238** - Split responsive webhelp into separate packages: for docs and for customers OPEN ,

✓ ~~**MTDOCS-1201**~~ - Explore possibility of having Cloudfront not cache default.html or index.html CLOSED

I want my content in production to be replaced in edge location cache immediately when I upload new content, is that possible?

Not really.

Content isn't cached in an edge location until it is viewed by someone local to that edge location. It could be that your content isn't in cache because it hasn't been viewed recently.

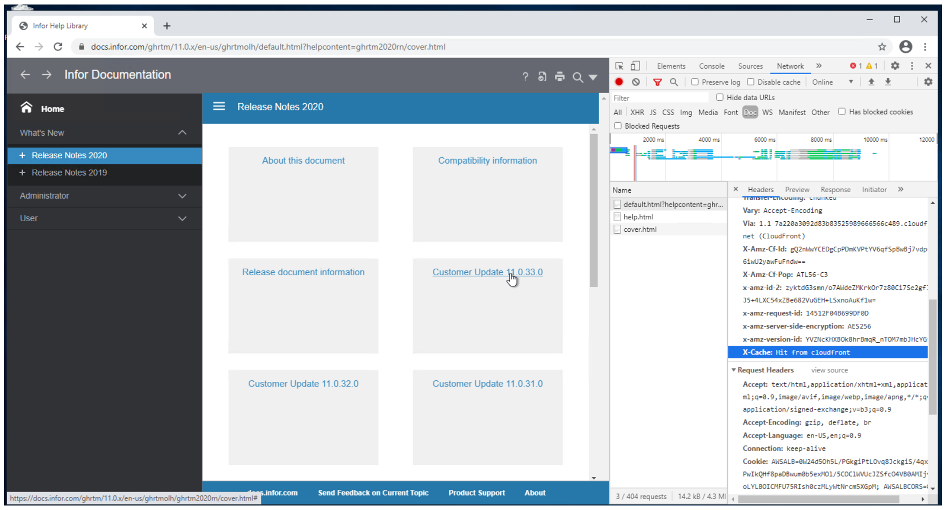
Once cached, there is no way to override the default TTL settings to expire particular content without permanently altering the content itself. For example, you can override default TTL settings on the CloudFront by setting headers on the content itself, but then the settings cannot be switched on and off in an automatic way. https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/Expiration.html

It is possible to invalidate content, which removes content from edge locations before it expires, but there are limits to invalidations. We are limited to 3000 files at a time, so would have to figure out some sort of batching mechanism, which would be different for each deliverable. But more prohibitive is the cost. We get up to 1000 free invalidations per month and then are charged for each invalidation beyond that. https://docs.aws.amazon.com /AmazonCloudFront/latest/DeveloperGuide/cloudfront-limits.html#limits-invalidations and https://docs.aws.amazon.com/AmazonCloudFront/latest /DeveloperGuide/Invalidation.html

So hopefully you can now see that it is not feasible to remove cached content from edge locations, especially when you know it will clear automatically within ~~24 hours~~ one hour or less.

How do I know if my content is coming from the S3 bucket or getting served from CloudFront cache?

If you want to do additional testing yourself, the information about whether the file you are viewing is coming from CloudFront cache (at the edge location) or from the actual source is captured in the Response Header details, which you can view in your browser's developer tools. Developer Tools > Network > filter on Doc or HTML file types, select the topic you are viewing in the list – this opens request details and you can see request and response info in the Headers tab, for example:



Infor Documentation

Release Notes 2020

About this document | Compatibility information | Release document information | Customer Update 11.0.33.0 | Customer Update 11.0.32.0 | Customer Update 11.0.31.0

```
Vary: Accept-Encoding
Via: 1.1 7a220a3092d83b83525989666566c489.cloudf
net (CloudFront)
X-Amz-Cf-Id: gQ2nWwYCEDgCpPDmKVPtYV6qfSpBwBj7vdp
6iwU2yawFuFndw==
X-Amz-Cf-Pop: ATL56-C3
x-amz-id-2: zyktdG3smn/o7AWdeZMKrkOr7z80Ci7Se2gf
J5+4LXC54xZBe682VuGEH+LSxnoAuKf1w=
x-amz-request-id: 14512F04B699DF0D
x-amz-server-side-encryption: AES256
x-amz-version-id: YVZNcKHXBOk8hrBmqR_nTOM7mbJHcYG
X-Cache: Hit from cloudfront
▾ Request Headers      view source
Accept: text/html,application/xhtml+xml,applicat
ml;q=0.9,image/avif,image/webp,image/apng,*/*;q
application/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Connection: keep-alive
Cookie: AWSALB=0W24d5Oh5L/PGkgiPtLOvq8JckgiS/4qx
PwIkQHf8paDBwum0b5exMOl/5COClWVUcJZSfcO4VB0AMIj
oLYLBOICMFU75RIsh0czMLyWtNrcm5XGpM; AWSALBCORS=
```

X-Cache: `Miss from cloudfront` – then the content was not cached, and the topic shown was retrieved from the website origin
X-Cache: `Hit from cloudfront` – then it is presenting the topic from an edge location close to you and it could be the older version – it will expire in 24 hours or less.
X-Cache: `RefreshHit from cloudfront` - RefreshHit means the edge location had the object, but thought it was stale, so it went to the origin (s3) to check staleness (Refresh). The origin responded that the object was indeed fresh and then cloudfront served the object from the edge location (Hit). The latency to the origin is still incurred in this case.

Does CloudFront caching have anything to do with browser caching?

No, this is completely separate. It appears that browser cache is an additional layer that we still need to remember. Browsers will cache files, like javascripts used for search, locally to try and help performance.

For example, if you wait a day and changes are still not displayed in your browser – or content works fine for you but not for a colleague, this is likely browser cache.

Recommendations remain the same for overcoming browser caching. Be sure to clear your browser cache, try using in private or incognito sessions. If things still look weird or not as expected, test content in your own local IIS to be absolutely sure and learn how to use Developer Tools to inspect cache information in response headers (see question above). Instructions on how to test your oxygen webhelp locally: Deliverables Collections User Guide#PublishonalaptopwithIISwebserverlaptopiis

Related JIRA: ✓ ~~DOCTOOLS-11064~~ - Oxygen Help - Unique filename for framework resources CLOSED

I don't recall CloudFront cache being such a pain before – what has changed?

The rollout of Secured Content functionality required us to put all static content (the entire contents of the S3 bucket which hosts all deliverables except Infocenters) behind CloudFront. Before the Docs v2.9.0 update, if URLs were for static content in the S3 bucket, the Apache web server took care of the request and directed it to the S3 bucket. Now, S3 is configured as a CloudFront origin and the only way to get static content from S3 is through CloudFront. Now only CloudFront is allowed access to our S3 bucket via a security role called Origin Access Identity. This change makes our content more secure because we no longer have a "public-facing" S3 bucket – which the Security Office and Infrastructure Ops teams have requested for a long while – and this is absolutely necessary for us to host secured content. But the consequence of this change is all website content accessed by you (or anyone near you) is cached in your edge location.

This comes into play when you as writer do the usual and natural process of verifying your content has been updated on the production site: normally you follow a view upload verify process: view the current collection or library on docs.infor.com (which caches the content), then upload changed content to docs.infor.com, then view your content again immediately to verify the change is there (but it's probably not there yet, because you are viewing cached content).

Is there any way to get around the caching so I can validate my content immediately on docs.infor.com?

Sorry, not really. See the "I want my content in production to be replaced in edge location cache immediately..." section above.

You can try to get around the edge location cache by testing from a different location – for example, try to use the remote server we use for uploads in Atlanta, or ask a colleague in a different region. However, remember that if it is also cached in their edge location in the ~~24 hours~~ one hour preceding your upload, they will also see the old, cached content until the cache clears in their edge location.

You can do your testing in stable and trust that a successful upload worked. The stable deployment is a production proxy and is as similar to production as it can be. If you uploaded your content to stable and it looks good, when you upload the same deliverable to docs.infor.com, it will work the same.

You can wait ~~24~~ ~~8~~ **one** hour and validate your content then.

We are exploring the possibility of enhancing the Admin Console so you can validate your content directly from the console – the idea there is to bypass CloudFront entirely. Nothing in the Admin Console is cached at edge locations. We don't know if it is possible yet, but this work is captured in: ~~MTDOCS-1019~~ - Admin console: Test content from content table (bypass CloudFront) CLOSED

My colleague and I are in the same region of the world, how is it possible that we are seeing differences in content?

You may find this explanation on stack overflow helpful.

TL;DR: CloudFront has many edge locations (where your browser connects and the content is physically stored). Some major cities have more than one edge location. Each request that hits cloudfront is routed to the "closest" edge to where you are.

The internal workings of Cloudfront are not public information, but the general consensus is edge locations don't share caches. For example you are in the midwest USA and your request routed through and was cached in Chicago. It's possible any request could hit any of the (currently) 6 edge locations in Chicago and until users generate hits of the same object in all 6 locations, the chances are the next request will be a miss. If I am "closer" to Minneapolis, MN and my request caches content there, since there is only one location in Minneapolis at the moment, the next request through Minneapolis is more likely to experience a hit from cloudfront cache.

This is not to say browser cache is not also a factor. So as stated above, it's a good idea to be sure to clear browser cache and use incognito or private sessions.