Mobile App Performance SDK

Configuration Guide

November 5, 2020

The Mobile App Performance (MAP) SDK

The Akamai Mobile App Performance software development kit (MAP SDK) helps you build enhanced mobile experiences for your Android and iOS users. The SDK effectively extends the Akamai Edge all the way to your mobile device allowing you to customize and deliver instant app experiences based on caching, network-awareness, and last-mile optimization technologies.

Once the SDK is integrated with a mobile application, the application can be deployed to the App/Play store normally. Audiences who download and begin using the application with the integrated SDK will automatically begin to experience a vastly improved experience as intelligent optimizations begin fine tuning the network layer of your application to improve overall app performance.

This document covers the features and configuration values available in the control panel.

Features of MAP SDK can be divided into three main groups: cache control, network optimization, and analytics.

Group	Feature	Description
Network Optimization	Network Awareness	Determines real-time network quality and makes it available via an API. Use knowledge of the current network conditions to tailor the app experience.
	SureRoute for Cellular	Two lookups race against each other to determine a primary and secondary path to the Akamai Edge. The winning path is used for subsequent requests for up to 20 seconds. Improves performance and reliability of requests.
	Adaptive Image Compression	Employs standard JPEG compression methods when slower network speeds are detected for a device. This is not configurable, but part of the MAP SDK.
	Adaptive Network Optimization	Optimizes the TCP profile to improve time to first byte and total download time of objects. This is not configurable, but part of the MAP SDK.
	Image Manager	Delivers images optimized for the device and viewing areas.

	Brotli Compression	Brotli can improve performance by reducing the number of bits transferred.
	QUIC beta	End to end implementation of the low-latency Internet transportation protocol.
Cache Control	Contextual Pre-Positioning	Proactively downloaded Images, videos, and other static content to the user's device when connectivity is available and congestion is low.
	Universal Cache	Browser-like caching within your mobile app (Images & API) to improve performance, reliability and to reduce network calls.
	Pre-Caching	Pre-fetch objects in the foreground based on Machine Learning and Analytics for better performance.
Performance Monitoring	Event Monitoring	Define User Experience centric events and monitor performance impact on App experience.
	A/B Testing	Turn features on and off for a percentage of users to measure the latency improvement, and to phase the rollout of enhancements, or set a percentage of users with all enabled features to compare against a holdout group of users.
	mPulse Plug-In beta	Plug In App performance data to your mPulse DPM Dashboard with no additional effort or SDK.

Cache Control

The MAP SDK allows you to cache content on a mobile device, increasing performance. By pre-positioning content on the device, and using browser-like caching, your mobile apps will not only perform better, but will provide an improved user experience even when network conditions are bad.

Pre-positioning Content

One of the capabilities of the MAP SDK is the ability to pre-position content on the device. Once this content is on the device, it's ready when the app user needs it, even if network connections are slow or non-existent. This creates a seamless experience for the user, regardless of connection speed or availability. While heavy content, such as images and videos, create the largest performance benefit, it makes sense to pre-position smaller items as well, such as a brand logo, or anything that's downloaded frequently.

There's a lot of flexibility into how and when content is pre-positioned. By segmenting audiences into different groups and associating content (URLs) with those segments, you can then choose to push appropriate content during off-peak hours and/or when connection speeds are fast. For example, you might call one user segment Women's Bargain Shopper, and it loads items from your flash sales and closeouts, with new images on a daily basis. Another segment might be called Men's Timeless Classics, which loads premium men's clothing, with a longer update interval.

Another clever way to pre-position content segments is by geography. For example, if you have multiple locations where maps, tickets, videos, and other content is specific to that location, you can pre-position the appropriate content on the user's device in that locale. When they change locations, you'd change the pre-positioned content on their device.

Configuration Settings

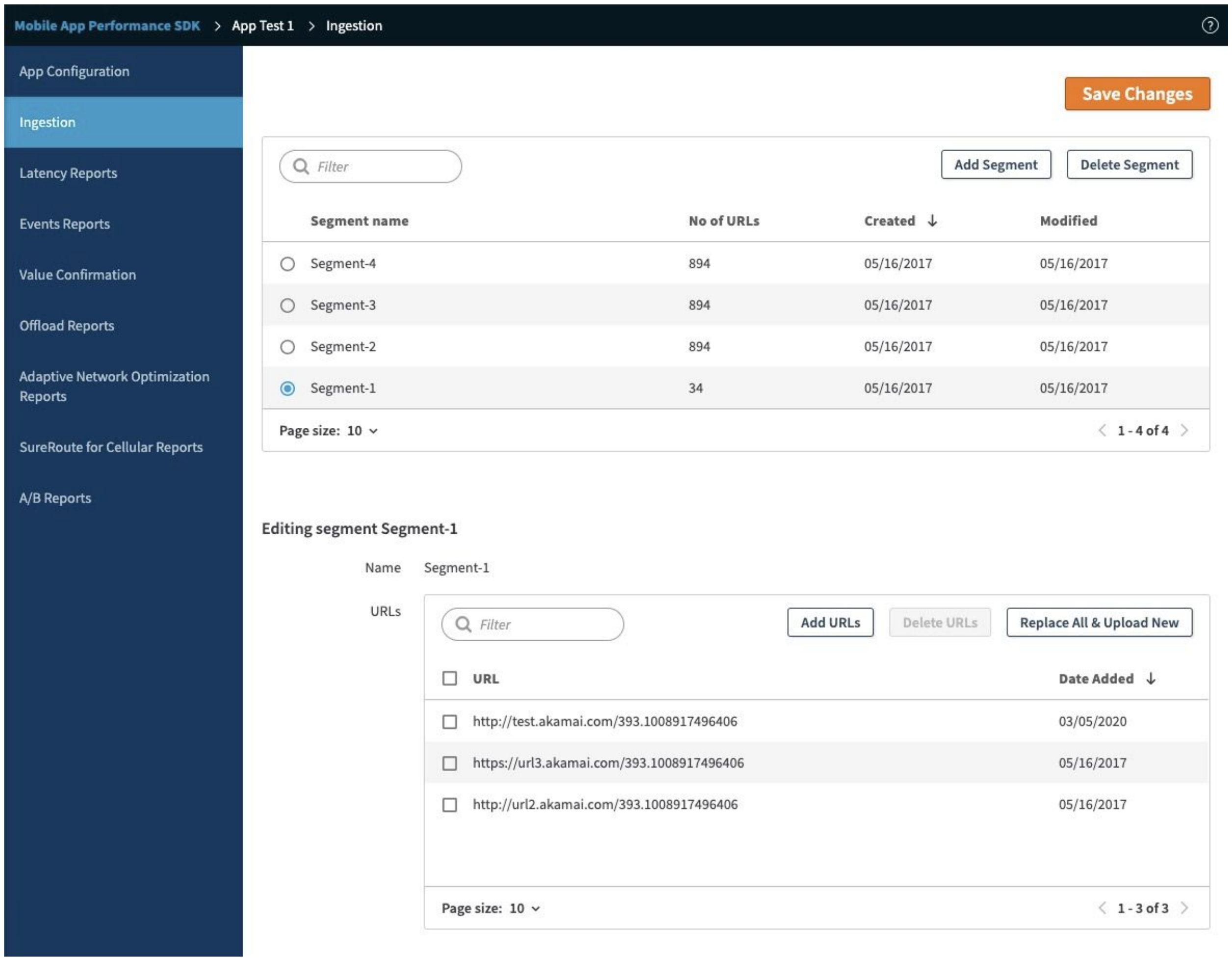
These configuration settings are available on the control panel.

Field	**Default**	**Action**
Pre-positioning	On	Enables and disables the feature, on or off.
Download only when	Wifi	Choose to download over wifi and/or cellular.
Daily Data Limits	1000 MB	Set as appropriate. 1000 MB is a default value, not a recommendation.

Usage

You can choose to download content over wifi and/or cellular, and at what time downloads occur. There aren't specific guidelines, but keep customer data plans in mind. If you enable Cellular downloads, it's advisable to set a very low daily data limit.

The daily data limits are default values provided, but are not recommendations. It's best that you limit downloads to wifi only.



Segment name	No of URLs	Created	Modified
Segment-4	894	05/16/2017	05/16/2017
Segment-3	894	05/16/2017	05/16/2017
Segment-2	894	05/16/2017	05/16/2017
● Segment-1	34	05/16/2017	05/16/2017

Name Segment-1

URL	Date Added
http://test.akamai.com/393.1008917496406	03/05/2020
https://url3.akamai.com/393.1008917496406	05/16/2017
http://url2.akamai.com/393.1008917496406	05/16/2017

Ingesting Content

In order to pre-position content, you need to associate groups of users with appropriate content (URLs). We call this process “ingesting content” and it’s typically done through the Ingest API. However, for initial setup, testing purposes, and to review already-ingested content, there’s a user interface tab dedicated to ingest.

Procedure:

1. Navigate to Mobile App Perf SDK.
2. Click **Add a New App**.
3. Add your information for your new app.
4. Click **Save**.
5. Once your app is created, you will be brought to the App Configuration tab. Click **Ingestion**.
6. From this tab, click **Add Segment**.
7. Name the segment, and add the URLs separated by commas or upload a CSV file.
8. Click **Add Segment** to finish the task.

Note: If you wish to add a segment to an existing app, from the **Add a New App** page, click the app you want to add to, and follow the steps starting at step five. You can also edit segments and their URLs, or delete and replace existing segments. Bulk operations are better done through the Ingest API.

Content Prioritization

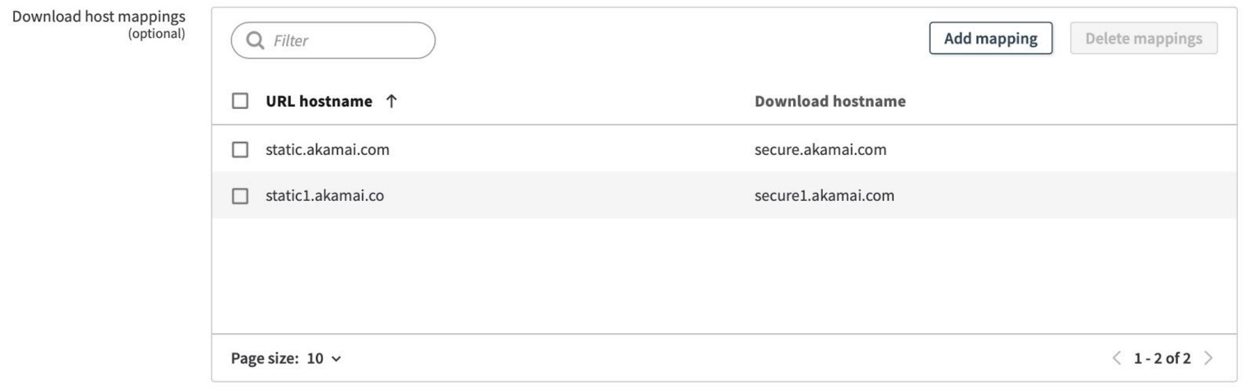
Segments can be assigned a priority between 1-100, with 1 being the highest. Multiple segments can have the same priority. Segments with no priority will be considered lower priority than segments with a priority. The URLs within a segment will be downloaded in the order they are listed.
Priority can be assigned during Segment creation or later in Ingestion settings.

Segment Level Controls

Segments can be configured to be downloaded regardless of device connection type (WiFi/cellular) or only when connected to WiFi. By default, segments use global pre-positioning settings.

Load Content from Alternate Source

Ingestion settings also provide an option to configure loading pre-positioned content from an alternate hostname. This example shows how you can choose to download content from Secure CDN rather than Static CDN during the pre-positioning.



URL hostname	Download hostname
static.akamai.com	secure.akamai.com
static1.akamai.co	secure1.akamai.com

Pre-Cache

Pre-cache helps MAP SDK learn the popular objects in demand across your user base for the mobile application and then starts downloading them to the device cache when the network is idle to enhance offload at the device.

Pre-caching studies network requests for the last seven days the first time when it has been configured and then starts updating the pre-cache segment once every 24 hours to stay up-to-date with changing content requirements and patterns. Pre-caching only works for objects/responses which can be cached and are defined using "Cache-Control" header with a valid "max-age" value.

When the device is offline, content responses for requests are served from the cache until the expiration date. However, if the application sets the “max-stale” value of the “Cache-Control” request header, MAP SDK will continue to serve a stale response even after the expiration date (“max-age”) from the cache.

If you are using pre-positioning, pre-cache, and Universal Cache together, any duplicate content is merged and its state reflects the ingested content.

Configuration Settings

The following configuration setting is available via the control panel.

Field	**Default**	**Action**
On/Off	On	Enables and disables Pre-cache.
Hostnames	Blank	Allows you to specify domains in your app for which pre-cache must be enabled. Protocol should not be included. Eg: “example.com” without “https://” or “images.example.com” for images and “api.example.com” for APIs.

Universal Cache

Universal cache allows you to effectively replicate the existing browser caching rules. The net effect is to give mobile apps the same benefits as other content, like reducing the number of requests to the network, offloading delivery costs, and improving performance for objects requested multiple times.

Universal cache honors the cache control headers that are being set on the server side by caching the content and storing it locally on disk. If the cache control header value expires and the content has changed, the content gets refreshed automatically the next time it gets requested by the host app.

When the device is offline, content requests are served from the cache until the expiration date. However, if the application sets the “max-stale” value of the “Cache-Control” request header, MAP SDK will continue to serve a stale response even after the expiration date ("max-age") from the cache.

If you are using both pre-positioning and Universal Cache, any duplicate content is merged and its state reflects the ingested content.

Configuration Settings

The following configuration setting is available via the control panel.

Field	**Default**	**Action**
On/Off	On	Enables and disables cache control.
Content to Refresh	All Content	Click **Submit Purge** when Universal Cache is selected to invalidate all content. If you want to go by specific URLs, either enter your list, or upload them with the Choose File button. This takes 5 minutes.
Refresh Method	Purge	If purging by URLs is selected then you have the option to invalidate or delete.

Purging Content

The current time between issuing a purge and the cache getting invalidated or deleted is 5 minutes.

The purge mechanism works two ways: invalidate or delete content from cache.

- **Invalidate** – The MAP SDK does an IMS GET request for the content specified in the cache before serving it to the app. The If-Modified-Since GET is done once per purge. Invalidate works at either the universal level or by URLs. With Invalidate, content can still be served from cache if the server indicates that it did not change.
- **Delete** – Deletes the content from the cache. Only works by URLs, and can increase the load on your origin server more than invalidate

Usage

Verify that cache control headers are set with a TTL that is effective in the future. For example, you could test with Charles Proxy or similar to verify that cache control headers are set the way you want. And then test again to see if the content is being cached within the SDK.
Whenever Universal Cache capability equals "false" (either its absolute value or logical value based on parent capability being turned on/off), SDK will purge all foreground downloaded content.

Limitations

To preserve system storage, files larger than 3 MB are not cached. Optimize your content so that it's smaller than this size.

Network and File Optimization

The MAP SDK improves mobile app performance through network optimization and file compression.

SureRoute for Cellular

SureRoute for Cellular performs two DNS lookups, by using the device's default resolver, and by using a DNS server such as OpenDNS or Answer-x. You can't configure the second DNS server through the control panel.

Issuing two requests improves the chances of a successful request through unreliable network connections or bottlenecks, and increases the chances of retrieving an object from the Edge cache. It's basically a race to see which lookup is faster.

The goal is to retrieve two separate IP addresses that make the same request. When the first bytes are retrieved from the winning path, the request to the losing path is cancelled. The winning path is then used for the next 20 seconds of downloads, or whatever the *stickiness interval* is set to, between 0-200 seconds.

When a request matches the URL pattern registered to run a race, the following steps occur:

1. Two DNS requests are sent on two separate DNS servers, as previously explained.
2. For each DNS resolver,
    a. We request A and AAAA records. We only request AAAA records if the interface supports IPV6, as we've found that lack of support can slow down requests.
    b. If the first response is an A record, we give the AAAA record a few milliseconds to respond as well.
    c. If there is an AAAA record, we use it to run the race because we typically see better performance this way, and because we want to respect the limited resources of the device.
    d. If there's no AAAA record then we use the A record to run the race.
    e. If IP can't connect for any reason, the next IP from DNS is tried.
3. A race is started on both paths if there are distinct IPs. If there aren't any distinct IP addresses, then they will not race.
4. Edge servers ensure that a second request is not sent to the origin, protecting against request amplification.

5. The winner is determined by time to first byte (TTFB) timing, and holds that IP address for that host name for a period called the *stickiness interval* (default 20 seconds).
6. Any URLs matching the pattern that triggered the race will automatically use the winning IP address for the stickiness interval without running a new race.

Typically the initial race is the same speed or slightly slower than the default single path, and then all the subsequent requests that match the pattern during the stickiness interval are typically 10% to 20% faster, but sometimes 30 to 40% faster.

Stickiness Interval

The stickiness interval is the maximum amount of time (in seconds) that the winning path is used. By default this is set to 20 seconds, and we recommend increasing this value up to 60 seconds.

Note: Setting the stickiness interval to zero seconds effectively runs a race one time and is used for a single download.

To illustrate this, here is an example of a SureRoute for Cellular "race" that is run for `x.foo.com/path/object.jpg`.

- If the stickiness interval is set to 60 seconds, then the same winning IP address will be used for up to 60 seconds.
- If a wildcard match is used, for example x.foo.com/path/.*\.jpg then any object in the path directory with a .jpg extension will use the winning IP address for up to 60 seconds to retrieve content.
- If the stickiness interval is set to 0 seconds, the race is run and used for a single download. The SDK won't keep the results to be used for subsequent requests that also happen to match the pattern.

Configuration Settings

The following configuration settings are available via the control panel.

Field	**Default**	**Action**
On/Off	Off	Enables and disables the feature.
URLs	Blank	Allows you to specify which URLs in your app that triggers the second request. Protocol must be included, and wildcards are allowed. Acceptable values include http https, and regex expressions.
Stickiness Interval	20 sec	Specifies the number of seconds for which the winning IP address is reused, and applies to the match that triggered

		the race. Valid values are 0 to 200 seconds. A value of 0 means that the SDK will never reuse the winning path.

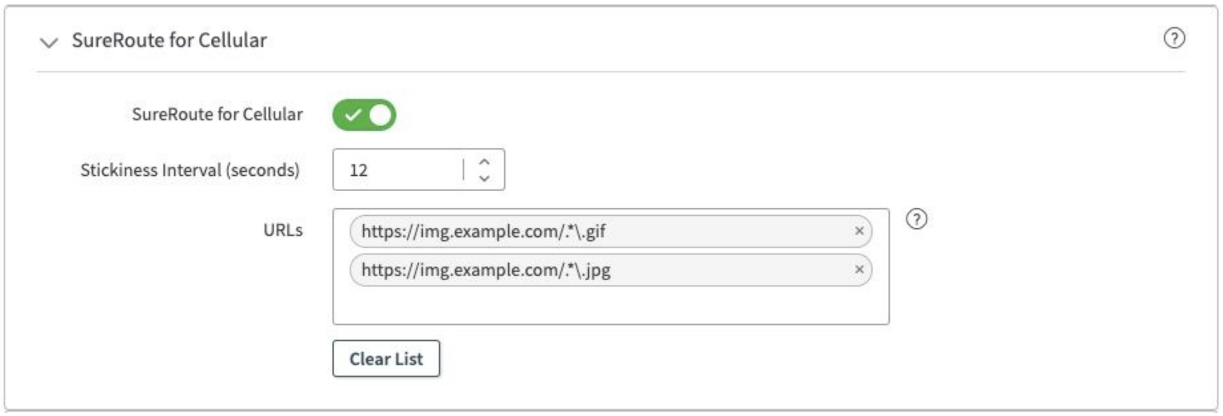
Usage

Ideally you'd use SureRoute for Cellular on the first object on each app view and have the winning path applied to all the objects within that view on the same hostname. This way you only run the race one time, and the benefits are applied to each of the objects within the view.

If you run races on every object, it will increase data usage of both your contract and the user's data plan, so use this feature sparingly and reuse race results by applying a stickiness interval to a wildcard match. Race results can be reused for up to 200 seconds.

URLs

You can specify which URLs in your app that triggers the second request by entering each one in the field provided. Include the protocol and end with a backslash as shown in the example . Regex expressions are allowed as wildcards.


SureRoute for Cellular

SureRoute for Cellular: ✓

Stickiness Interval (seconds): 12

URLs:
https://img.example.com/.*\.gif
https://img.example.com/.*\.jpg

DNS lookups for different domains can resolve to different maps and IP addresses, so don't run races on just the hostname. For example, if you have host.com with three subdomains, News (`news.host.com`), City (`city.host.com`), and Hotel (`hotel.host.com`), separate the SureRoute for Cellular races across all subdomains.

Using Regex and Wildcards

The SDK supports regex pattern matches in the URL list. We won't document regex here, but here are some regex examples you can use in the URL path.

To match a specific URL
To match a specific URL, end with $. '$' ends the pattern to prevent sub-paths from matching.

For example: `https://developer.akamai.com/tools/map/$`

To match a URL with trailing parameters
To match a URL with trailing parameters, omit the $.

```
https://www.akamai.com/us/en/multimedia/images/intro/get-started-home
-intro.jpg
```

This leaves the pattern open for query parameters such as the following:

```
https://www.akamai.com/us/en/multimedia/images/intro/get-started-home
-intro.jpg?imwidth=1366
```

To match a path recursively, with all files and sub-paths
Omit the trailing $ to match a path with all files and sub-paths under it:

```
https://www.akamai.com/us/en/products/web-performance/
```

If your URL includes a question mark, then that must be escaped with a backslash ('\'). For example,
```
https://www.akamai.com/us/en/multimedia/images/intro/get-started-home
-intro.jpg\?imwidth=
```

will match
```
https://www.akamai.com/us/en/multimedia/images/intro/get-started-home
-intro.jpg?imwidth=1600
```

Best Practices

- SureRoute for cellular must only be configured for domains that are using Akamai ION for content delivery.
- **Only use cacheable objects. s**Using non-cacheable objects will cancel the secondary request.
- The more URLs you use, the better.
- The test object should be the first object requested.
- Have one test object per app view. For example, if you have an e-commerce app that loads large images at the beginning of the Home view, Orders view, Departments, etc., you'd want to run a race for each app view.

Limitations

The current implementation of SureRoute for Cellular for iOS will not work on SNI slots. Ensure an SNI slot is not in use to take advantage of this capability. This limitation does not apply to

Android. This limitation corresponds with a limitation for the use of QUIC (Adaptive Network Optimization).

Network Awareness

Network Awareness monitors the network quality that a user is experiencing in the app. The quality value (POOR, GOOD, EXCELLENT) is then provided via an API so that a developer can tailor the user experience appropriately. The network quality value is also passed by the SDK in the mobile connectivity header where it can be utilized by Edge servers or the customer's origin servers to tailor the experience.

Configuration Settings

The following Network Awareness settings are available via the control panel.

Field	Default	Action
On/Off	On	Turns the feature on or off.
Bandwidth Settings	Off	Allows you to change the bandwidth settings. Generally, you would keep the default setting, unless you are an advanced user.

Security Considerations

If you have Edge security features enabled, they may strip off the mobile connectivity header before it can be applied to image compression or reach the origin servers. Check with your PS consultant if you have Edge security features enabled.

Adaptive Image Compression (AIC)

Adaptive Image Compression (AIC) utilizes Network Awareness to employ optimal JPEG compression based on the quality and performance of the network. By evaluating the latency between the Akamai Edge platform and the application running on the device, MAP SDK is able to make decisions to apply greater image compression to maintain a consistent user experience.

AIC is On by default. **To disable AIC:**

1. Navigate to Mobile App Perf SDK, and open the app that you wish to update.
2. From the App Configuration tab, expand **Other**.
3. Toggle the Adaptive Image Compression option off.
4. Click **Save Changes**.

Note: AIC is automatically optimized by the mobile-connectivity header values, while Image Manager is not.

AIC cannot be enabled if Image Manager is being used. These two features are not compatible.

Usage

AIC delivers objects that are more suitable to reported network conditions. Having the network conditions as part of the analytics record allows for a more accurate assessment of the effectiveness of AIC.

‘tpResult’ is part of the URL analytics in the analytics upload message. tpResult : {0,1,2} where 0=excellent, 1=good, and 2=poor based on network quality pings.

Adaptive Network Optimization (ANO)

Adaptive Network Optimization speeds up mobile apps by reducing TTFB, total download time (TDT), and packet loss.It improves throughput for mobile app requests. ANO accomplishes this by providing client-side data to the Akamai Edge Server, which is used to select the best congestion-control algorithm and configuration settings between the device and the server.

Akamai’s Edge Servers selects the optimal choice between Transmission Control Protocol (TCP) or Quick User Datagram Protocol (QUIC) based on historical data as well as real-time factors. These factors include the device type, network connection type, carrier, and the real time and historical network conditions provided by the device.

Transport layer protocols include TCP and QUIC, and congestion control algorithms currently include Fast, New Reno, BBR among others. Algorithms are further optimized by applying profile-based settings. Profiles can range from conservative to super-aggressive.

- A conservative profile might be selected based on a lower quality device type, a carrier with poor historical network quality, or poor current network conditions or signal strength. This profile might start with a very small Initial Congestion Window (ICW) and then increase very gradually as packets are successfully sent and received.
- Under great conditions, on a high-end device, and with a high-quality carrier, an aggressive profile might start with a much wider ICW and react more slowly as the packet queue increases.

ANO is a key optimization capability of MAP SDK. It is configured automatically and adapts to changing conditions. This means that it is on by default, is used for all communications between the device and the Edge (barring any A/B testing control groups), and it does not require manual configuration.

Limitations

The QUIC protocol capability of ANO utilizes SNI slots, and the current implementation of SureRoute for Cellular for iOS will not work on SNI slots. If your iOS app has QUIC set to On and the Cronet Library integrated, turn SureRoute for Cellular Off. This limitation does not apply to Android.

Brotli Compression

The Android Brotli library allows your Android app to accept Brotli headers and apply content encoding. Testing shows a 15-20% improvement in performance over GZIP.

For iOS apps, Brotli is supported by the OS with version 11.0. However, the MAP SDK does not have any effect on IOS for this feature.

Brotli is On by default. **To disable Brotli Compression:**

1. Navigate to Mobile App Perf SDK, and open the app that you wish to update.
2. From the App Configuration tab, expand **Other**.
3. Toggle the Brotli option off.
4. Click **Save Changes**.

Image Manager

Image Manager optimizes images based on the characteristics of the device and viewing area. By enabling Image Manager, you'll get better images and better performance in native mobile apps. Image Manager is disabled by default, but can be enabled on the same screen as Brotli.

To take advantage of this feature, have Image Manager enabled and configured for web properties in Akamai Control Center. You can sign up for a trial of Image Manager from the Marketplace on Akamai Control Center.

Remote Notifications

MAP SDK uses silent push notifications to sync pre-positioning content and SDK configurations in the background. If the app does not have a need for pre-positioning content in the background, then this step is optional.

Usage of Firebase Cloud Messaging (FCM)

MAP SDK uses Firebase Cloud notifications to mainly sync pre-positioning content and SDK config in the background. The notifications leveraged by MAP SDK through FCM are silent which means that the end user is not notified or interrupted.

In order for FCM background notifications to work, follow the Firebase Messaging integration guide as documented at Google Firebase website. After Firebase is integrated in the app, complete the following steps:

- Paste the FCM server key in a plain text file and save it with the name key.txt.
- Once saved, upload the file to MAP SDK portal in the “Google FCM Key” field on MAP Portal as shown below. The API Key is then used by MAP Control Server to trigger map related silent notifications - for pre-positioning any content configured for the app.

The app needs to provide hooks in its implementation of its FirebaseMessagingService to pass the messages to SDK. Instructions on making supporting code changes are documented in our Android integration Guide for MAP SDK.

Analytics and Reporting

The MAP SDK provides the capability to measure both standard performance events, such as TTFB and TDT, as well as custom events within the app.

Reports

The MAP SDK library also collects network-related statistics while serving content. These include HTTP TTFB, request size, response size, duration, and others. These stats are periodically sent to the MAP SDK server for access via the Web portal. This information can be used to enhance the user experience by taking necessary actions based on network state.

Latency Reports

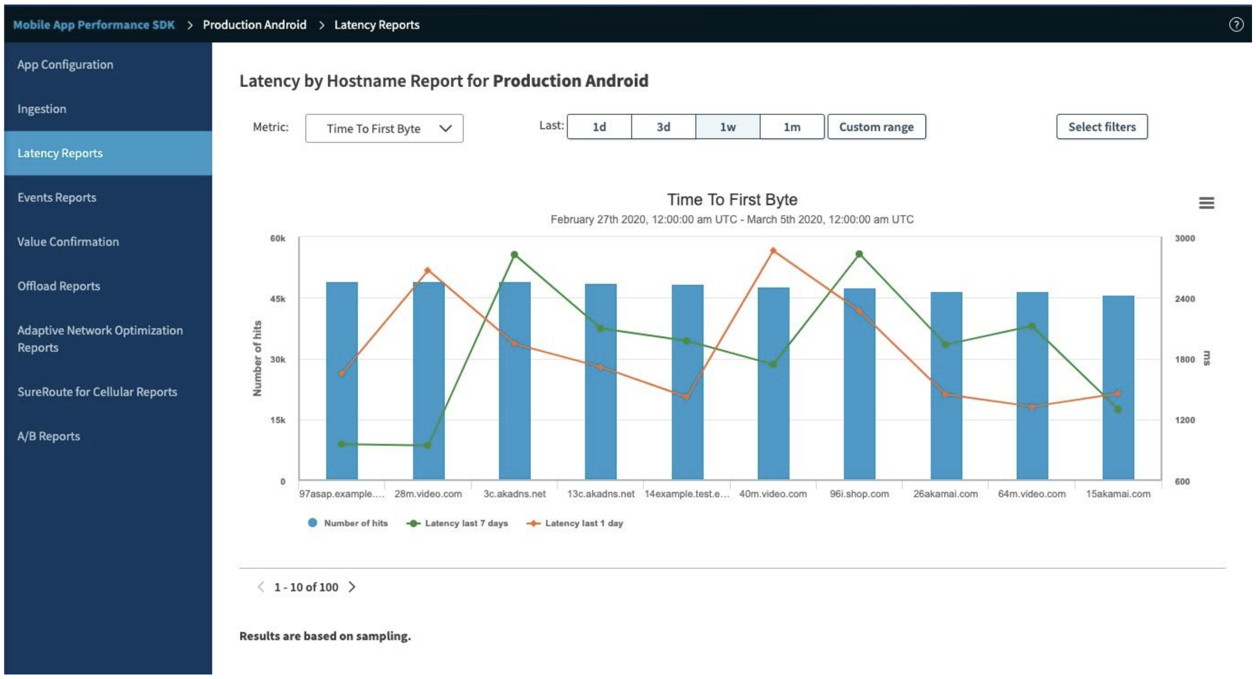
The Latency Reports tab shows reports based on standard performance events, such as TTFB and TDT.

To view latency reports:

1. Navigate to Mobile App Perf SDK, and open the app that you wish to work with.
2. From the **Latency Reports** tab, select the type of metric and the time frame you want to display in the report.
3. Additionally, filters can be added to further refine the report data.

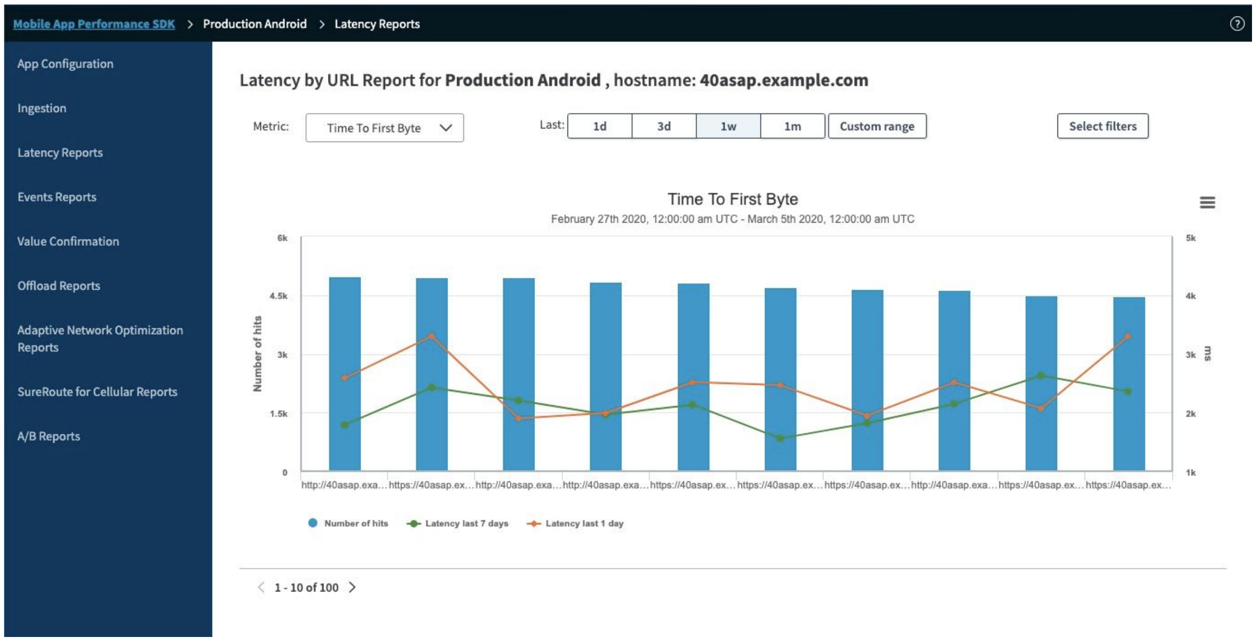
4. Click **Apply Filters.**

The initial report gives you an overview of all hostnames, sorted by number of hits.



View all URLs in a Hostname

You can go deeper into a hostname to inspect all of its URLs. Click one of the bars that represents a hostname and you can see the associated URLs.

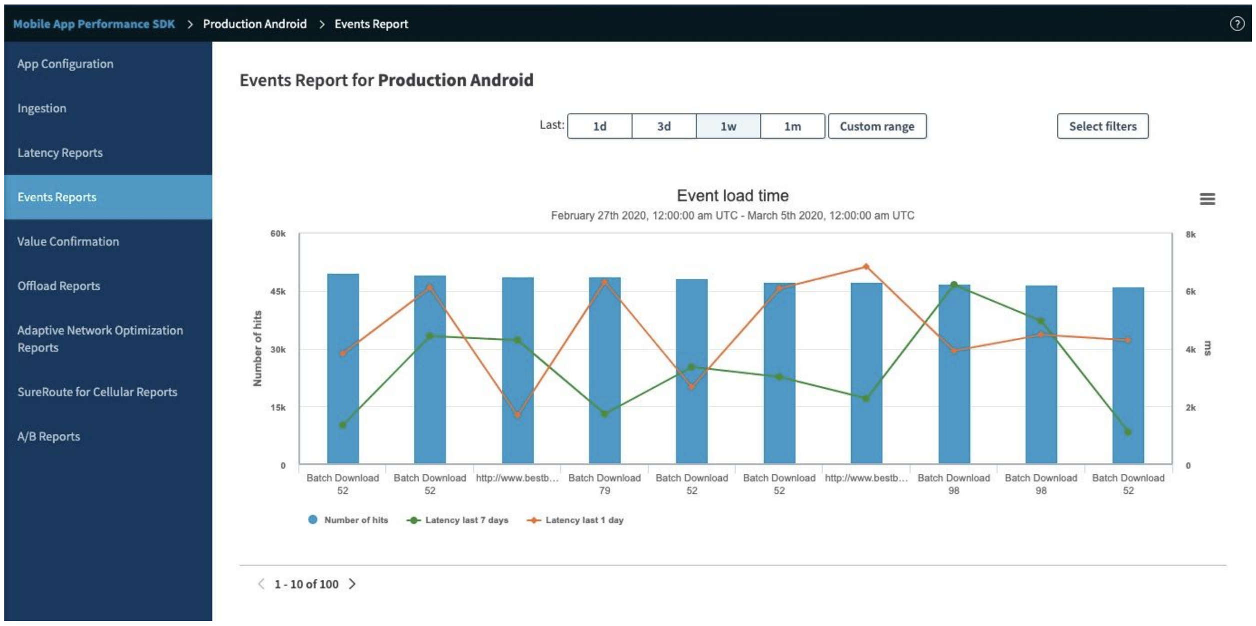


Mobile App Performance SDK > Production Android > Latency Reports
App Configuration
Ingestion
Latency Reports
Events Reports
Value Confirmation
Offload Reports
Adaptive Network Optimization Reports
SureRoute for Cellular Reports
A/B Reports
Latency by URL Report for Production Android , hostname: 40asap.example.com
Metric: Time To First Byte
Last: 1d 3d 1w 1m
Custom range
Select filters
Time To First Byte
February 27th 2020, 12:00:00 am UTC - March 5th 2020, 12:00:00 am UTC
Number of hits
ms
http://40asap.exa…
https://40asap.ex…
Number of hits
Latency last 7 days
Latency last 1 day
1 - 10 of 100

Events Reports

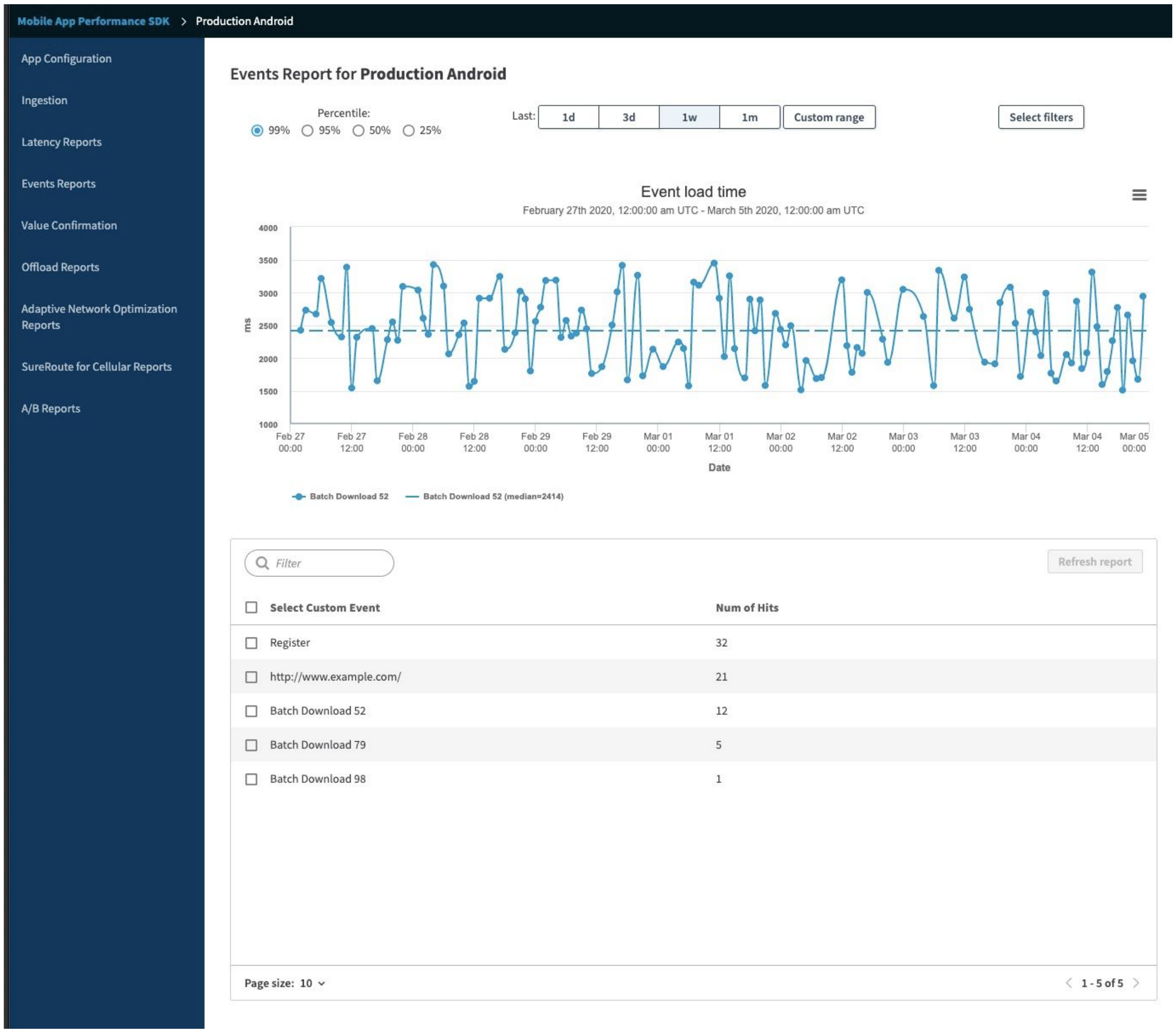
The Events Reports tab shows reports based on custom event timers that are set up to measure how long it takes for a user to execute app specific use cases. For example, the time it takes to navigate to a product or check out. If you haven't set up event reports, you won't see any on this tab. Please refer to the Integration Guides in order to set up the custom events.

When you select Events Reports, you'll see the Top Custom Reports by default. This report gives you a high level overview of your mobile app custom events, and allows you to quickly see if the performance for the past day is in line with historical performance.



Each vertical bar represents a custom event, with the bar height representing the number of hits. The orange and green lines show the latency in the last day and week, respectively. If you have more than 10 Custom Reports, pagination is enabled.

From the report overview, you can also go down into each specific custom report.


Mobile App Performance SDK > Production Android

App Configuration
Ingestion
Latency Reports
Events Reports
Value Confirmation
Offload Reports
Adaptive Network Optimization Reports
SureRoute for Cellular Reports
A/B Reports

Events Report for **Production Android**

Percentile: [✓ 99%] 95% 50% 25%

Last: 1d 3d [✓ 1w] 1m Custom range

Select filters

Filter

Refresh report

Select Custom Event	Num of Hits
Register	32
http://www.example.com/	21
Batch Download 52	12
Batch Download 79	5
Batch Download 98	1

Page size: 10

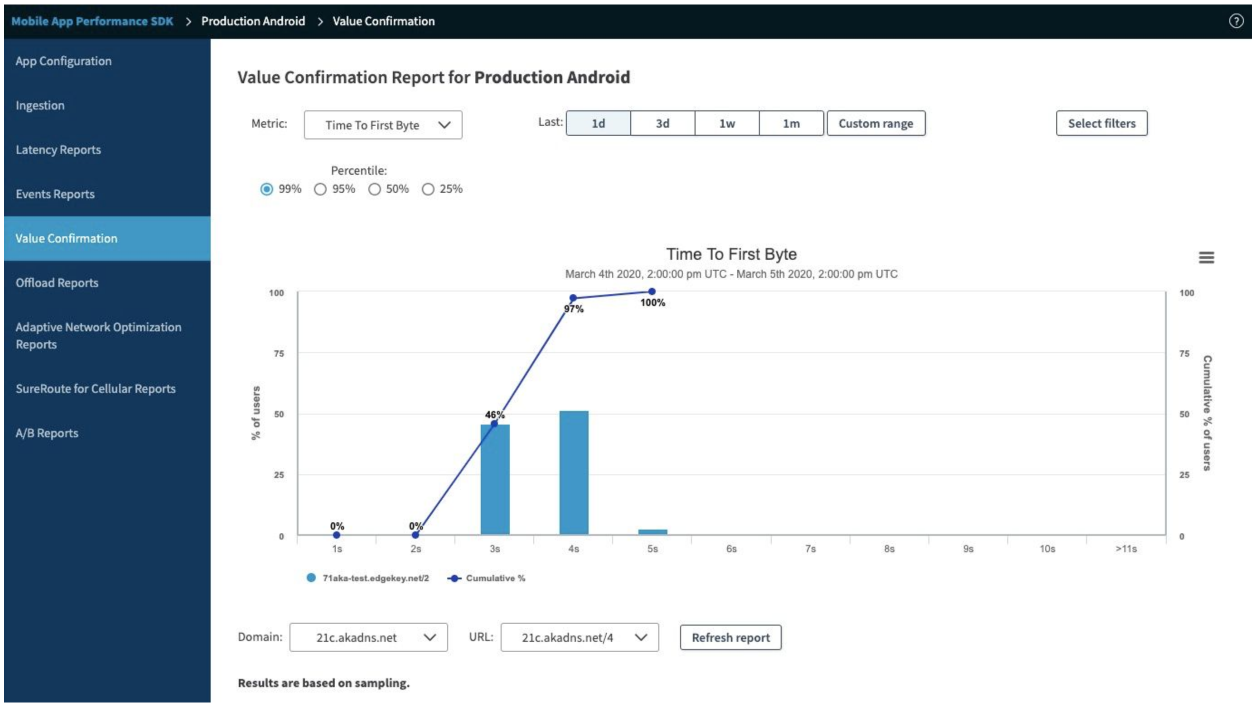
1 - 5 of 5

To view custom event reports:

1. Navigate to Mobile App Perf SDK, and open the app that you wish to work with.
2. From the **Events Reports** tab, you can sort reports alphabetically if you're looking for one by its name By default, the reports are sorted by number of hits..
3. Choose the Event Report that you're interested in. You can also multi-select to compare multiple events at the same time.
4. Click **Refresh report.**

Value Confirmation Reports

The final report you can view is one we call Value Confirmation, which gives you an overview of how long your app is taking to load. It shows you the percentage of users that are loading your app in 1 second, 2 seconds, and so on.


Mobile App Performance SDK > Production Android > Value Confirmation
App Configuration
Ingestion
Latency Reports
Events Reports
Value Confirmation
Offload Reports
Adaptive Network Optimization Reports
SureRoute for Cellular Reports
A/B Reports
Value Confirmation Report for Production Android
Metric: Time To First Byte
Last: 1d 3d 1w 1m Custom range
Select filters
Percentile: 99% 95% 50% 25%
Time To First Byte
March 4th 2020, 2:00:00 pm UTC - March 5th 2020, 2:00:00 pm UTC
% of users
Cumulative % of users
100
75
50
25
0
0%
0%
46%
97%
100%
1s
2s
3s
4s
5s
6s
7s
8s
9s
10s
>11s
71aka-test.edgekey.net/2
Cumulative %
Domain: 21c.akadns.net
URL: 21c.akadns.net/4
Refresh report
Results are based on sampling.

To view value confirmation report data:

1. Navigate to Mobile App Perf SDK, and open the app that you wish to work with.
2. From the **Value Confirmation** tab, select the type of metric and the time frame you want to display in the report.
3. Additionally, filters can be added to further refine the report data..
4. Click **Refresh report.**

Offload Reports

Offload reports help to measure the efficiency of caching at the app level using MAP SDK. Offloading responses or objects within the app cache helps to reduce network overhead costs. It helps end users have a more seamless experience by not having to download response payloads repeatedly.

We recommend developers and app owners to review these reports at regular intervals to help validate if caching within the app is working as expected. Anything below 15% offload in volume or hits, is considered poor in-app caching, and should always be investigated and improved.

Here are the stats displayed in this report.

- List of domain names where requests are being made from the app.
- **Total Hits.** Total number of hits or requests being made from the app towards each domain name.
- **Cache Hits.** Total number of hits or requests that were served from the cache instead of traveling over the WiFi or Cellular network. ()

- **Cache Hit Rate.** Percentage of cache hits by overall hits.
- **Total Vol. (GB).** Overall traffic generated by app users in gigabytes.
- **Cache Vol. (GB).** Volume of traffic served from the cache instead of traveling over the WiFi or Cellular network.
- **Cache Vol. Rate.** Percentage of cache volume in gigabytes by the total volume in gigabytes.

The default date selection for this report is for the last 7 days or “1w” as seen in the offload stats table. However, users can change this or use the “Custom range” date picker to select any date range.

Offload reports carry data for the last 30 days at any given time. Events older than 30 days are automatically purged by the system to help maintain databases.

You also have the ability to filter this report by Cache Type, Platform, App Version or SDK Version to understand patterns and associated trends. The Cache Type is the feature responsible for caching content. Similar filters are available with all MAP SDK reports allowing you to divide data to be tailored to your needs.

As you continue through the Offload Report page, you’re able to get more insights on the URL level for each domain selection.

The URL insights are further divided into more options.

- **Served from Cache**. Requests that were served from SDK cache instead of traveling over the WiFi or Cellular network.
- **NOT Served from Cache (NOT Cacheable)**. The requests listed here either don’t have cache control headers or are larger than 3 MB. Developers and app owners can review these to make sure that no cacheable content is passed down as non-cacheable which would lead to unwanted network overheads for end users and the origin.
    - Note that MAP SDK does not cache any objects or responses larger than 3 MB even if it comes with cache control headers enabled. This is to make sure that end user device storage is not overwhelmed.
- **NOT Served from Cache (Cacheable) -** The requests listed here have a valid cache control header and are less than 3 MB. However, they were not served from cache. This happens when users are requesting an object for the first time. Developers and app owners review the insights listed here to figure out most popular objects and download to the device proactively by using Pre-caching or Pre-positioning.

These insights can also be exported as CSV.

SureRoute for Cellular Validation Report

SureRoute for Cellular performs two DNS lookups, one using the device's default resolver, and the other using a DNS server such as OpenDNS or Akamai's AnswerX.

Issuing two requests improves the chances of a successful request through unreliable network connections or bottlenecks.It increases the chances of retrieving an object from the Edge cache. It's a race to see which lookup is faster.

The SureRoute for Cellular Validation Report helps to validate the effectiveness of SureRoute for Cellular by domain, hits, and the percentage of benefit by finding a faster path.

Developers and app owners review these reports at regular intervals to help validate if SureRoute for Cellular is optimizing traffic with the expected benefit rate. Anything below 30% in the benefiting rate should be considered as an area of improvement. If you see a rate below 30%, refer to Best Practices for SureRoute for Cellular and reinspect your applied path matches and stickiness interval.

You can export insights about the URLs that were leveraged to run races along with the list of URLs which benefited from the race result

Adaptive Network Optimization Validation Report

Adaptive Network Optimization is a key optimization capability of MAP SDK. It is configured automatically and adapts to changing conditions**.** This setting is on by default, because it is used for all communications between the device and the Edge (barring any A/B testing control groups), and doesn't require manual configuration. However, this feature only works for domains that are leveraging Akamai ION for first- and middle-mile optimization.

ANO speeds up mobile apps by reducing the time to first byte, total download time, and packet loss. It improves throughput for mobile app
requests. ANO can do this by providing client-side data to the Akamai Edge Server, which is used to select the best congestion-control algorithm and configuration settings between the device and the server.

Transport layer protocols include TCP and QUIC. Congestion control algorithms currently include Fast, New Reno, BBR among others. Algorithms are further optimized by applying profile-based settings. Profiles can range from conservative to super-aggressive**.**

Akamai's Edge Servers selects the optimal type of TCP or QUIC based on historical data as well as real-time factors such as the device type, network connection type, carrier, and the real-time and historical network conditions provided by the device.

Because this is an automatic feature, it requires very little maintenance. However, developers and app owners can review these reports at regular intervals.

Connect with the community on https://community.akamai.com/,and post questions or suggestions.

A/B Testing

To better understand how the various features affect your users' experiences, you can toggle features on or off for a percentage of your audience. By applying different SDK capabilities to a certain percentage of users, you can

- compare feature performance against a holdout group, and
- roll out new features to fewer users and ramp up its use over time.

Set the percentage for the MAP feature set to be enabled, and the SDK automatically assigns the A and B groups.

The purpose of A/B testing is to show the value of using MAP features vs. not using any MAP SDK features. Any MAP SDK features that you choose to configure will be enabled ON for a group of devices (Group A) and will be disabled OFF a second group of devices (Group B). For example, pretend you set the percentage of devices with all features On at 80%. The SDK then turns all configured features on for 80% of the users, creating an A group, and off for 20%, creating the B group. This allows you to test and validate MAP SDK feature set's performance vs regular traffic which is unoptimized.

Note that the groups are reassigned whenever the device receives a new configuration, even if the percentages have not changed. Applying a new configuration effectively starts a new test.

The recommended value is to set the A/B testing to 80% initially, to validate the performance and reliability value of the enabled features. This allows a holdout group of 20% of devices who have no features enabled.

Configuration Settings

To access the configuration setting for A/B testing

1. Navigate to **Mobile App Perf SDK**, and open the app that you wish to work with.
2. From the **App Configuration** tab, click **A/B Configuration** to expand the settings.

These are the configuration settings available via the control panel.

Field	**Default**	**Action**
A/B Testing	On	Enables or disables the feature.

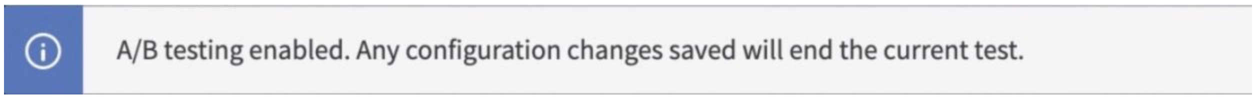
Set % of devices	80%	Choose to set all features on for a percentage of users. Universal A/B testing is on by default.

Device A/B Group Selection

The SDK determines groups A and B randomly based on the percentage specified in the configuration UI. A device is assigned to group A (feature on) or group B (feature off) when the SDK is initialized for the first time, and upon every subsequent configuration update. When configuration changes are made in the UI, it may take a few days before the A/B test results have enough samples to report accurate results.

Considerations

It's important to note that changes to configuration while A/B testing is enabled will reset tests. You'll see a warning when A/B testing is turned ON and when a change in configuration is made.



A/B testing enabled. Any configuration changes saved will end the current test.