Mobile App Performance SDK

Configuration Guide

July 30, 2018

The Mobile App Performance (MAP) SDK

The Akamai Mobile App Performance software development kit (MAP SDK) helps you build the best possible mobile experience 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 into a mobile application, that application is deployed into the app store normally. Audiences that 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. It's assumed that you've already integrated the MAP SDK with your mobile app.

The MAP SDK features can be grouped into three main buckets: cache control, network optimization, and analytics.

Cache Control

- **Pre-positioning** – Images, videos, and other static content is downloaded to the user's device when connectivity is available and congestion is low.
- **Universal Cache** – Browser-like caching within your mobile app to improve performance and reliability and reduce network calls.

Network and File Optimization

- **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.
- **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.

- **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.
- **Brotli Compression** - Brotli can improve performance by reducing the number of bits transferred.
- **Image Manager** - Delivers images optimized for the device and viewing areas.

Analytics and Reporting

- **Analytics** – Gathers network performance statistics from real user app usage. Automatically understand the time to first byte and total download time for requests (e.g. API, images, etc.). Or create custom events to ensure your most important app features are not impacted by network latency.
- **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.

The following sections examine these features in more depth, and how to configure them.

Cache Control

The MAP SDK allows you to cache content on a mobile device, greatly 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 a good user experience even when network conditions are bad.

Pre-positioning Content

One of the most powerful 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 video, 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.

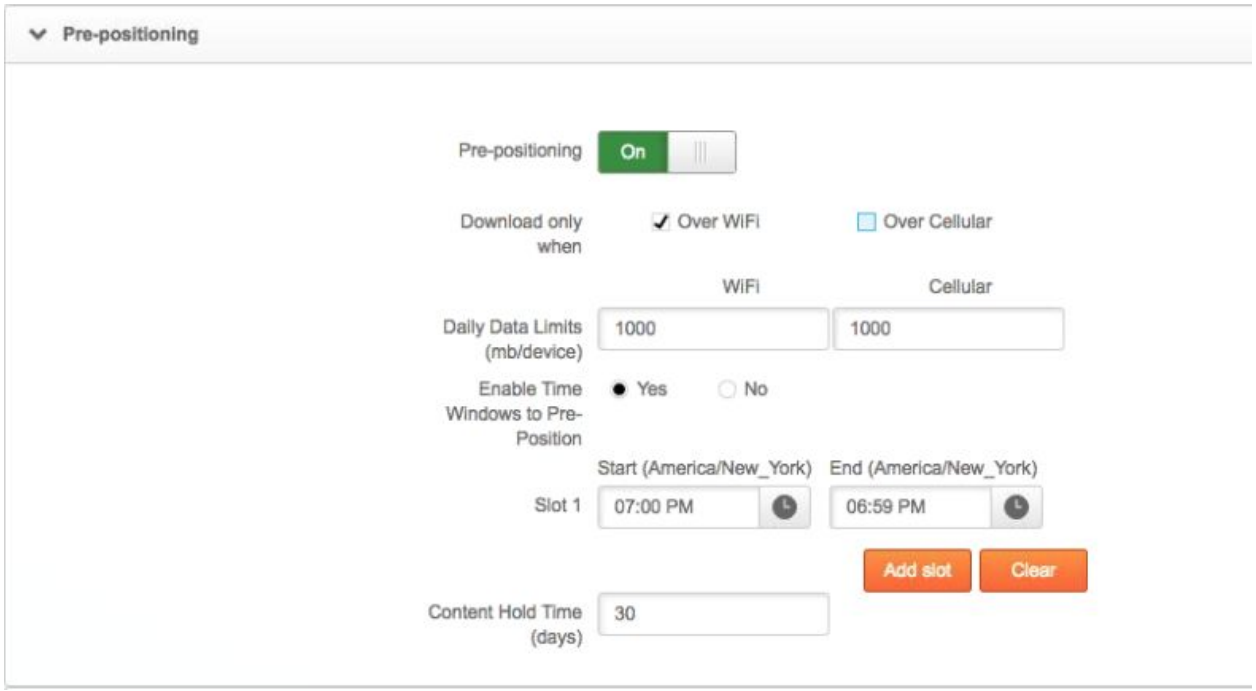
For example, you might call one user segment Women's Bargain Shopper, and it loads items from your flash sales and closeouts, with new images on a daily basis. Another segment might be called Men's Timeless Classics, which loads premium men's clothing, with a longer update interval. Another clever way 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

The following configuration setting is available via 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.

Enable Time Windows	N/A	Adjust the download time for off-peak hours of your users. You can add multiple slots.
Content Hold Time	30 days	Sets the maximum amount of time the content will be available on the device.



Usage

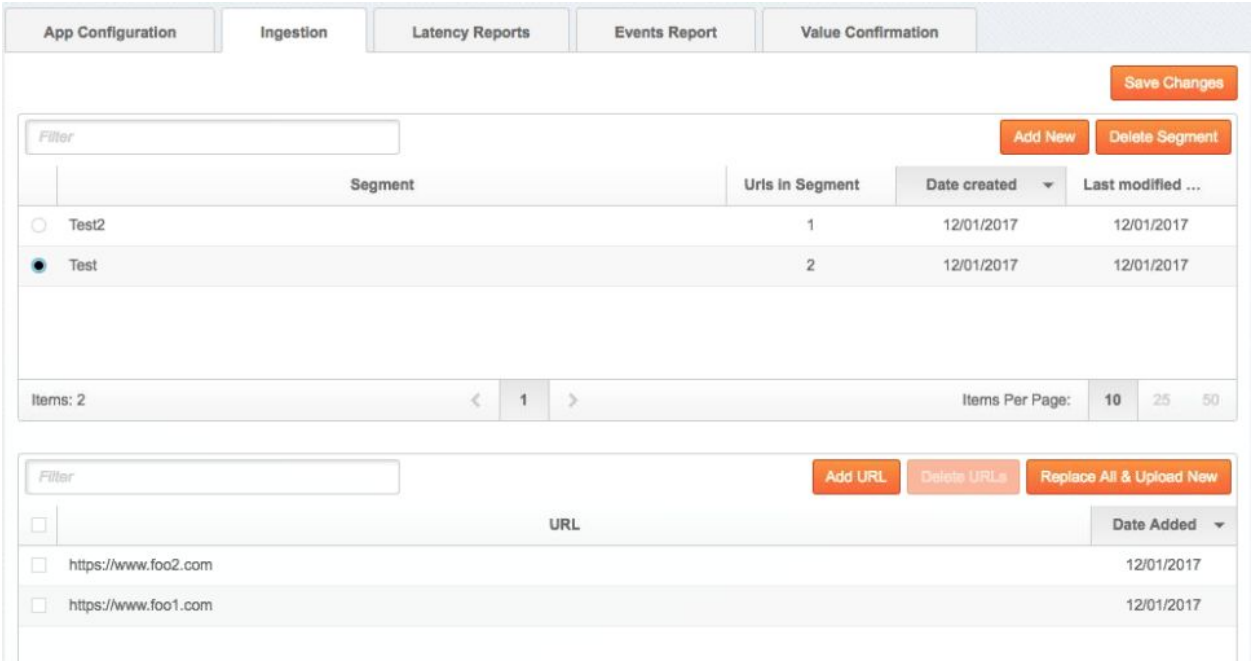
You can chose to download content over wifi and/or cellular, and at what time downloads occur. We don't have specific guidelines here, but common sense says not to blow up the customer's data plan. If you enable Cellular downloads, it's advisable to set a very low daily data limit.

The daily data limits are simply default values we've provided, and not recommendations. Until you have reason to do otherwise, it's probably best that you limit downloads to wifi only, and schedule them during off-peak hours.

Note that you can set multiple download windows if required. You can also set the maximum amount of time the content will be available on the device. After this duration, the SDK will no longer serve up this content, even if the device is offline. This means that if you have a limited-time offer, say a 30-day sale, and set the Content Hold Time to reflect that, the content will expire after 30 days whether or not the device goes back online or not.

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. In Luna, within the Mobile App Perf SDK, click the **Ingest** tab.
2. Click **Add New** and name the segment.
3. Add URLs separated by commas, or upload a CSV.

Through this same user interface you can view and edit existing segments and their associated URLs, or complete delete and replace an existing segment. Note that bulk operations are better done through the Ingest API.

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: reduce the number of requests to the network, offload delivery costs, and improve performance for objects requested multiple times.

Universal cache honors the cache control headers that are being set server side, caching content and storing it locally on disk. If the cache control header value expires, and if 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, and will serve stale content offline if that header is present. For example, if the application sets the request header as "Cache-Control", "max-stale", it will be served offline from cache even if the content is expired.

Note that if you're using both pre-positioning and Universal Cache, any duplicate content is merged and its state reflects that of ingested content.

Configuration Settings

The following configuration setting is available via the control panel.

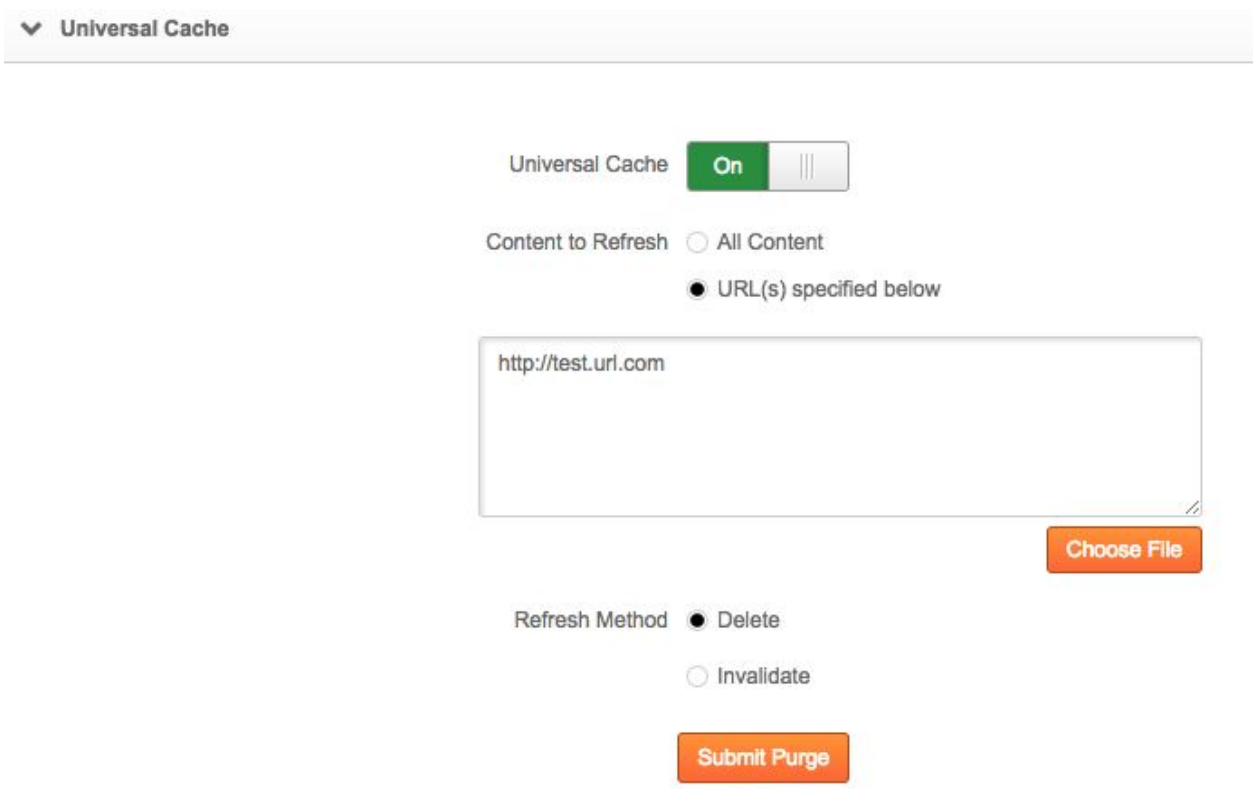
Field	Default	Action
On/Off	On	Enables and disables cache control, on or off.
Content to Refresh	All Content	Click **Submit Purge** to invalidate all Universal Cache content if all content selected, or invalidate or delete by URLs if URL(s) specified below is selected. This takes 5 minutes.
Refresh Method	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 contents from the 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

It's a good idea to verify that cache control headers are set with a TTL effective in the future. For example, you could test with Charles Proxy or similar to verify that cache control headers are set as desired. 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. You may want to 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: one using the device's default resolver, the other using a DNS server such as OpenDNS or Akamai's Answer-x. Akamai doesn't allow you to configure the second DNS server through the control panel, but we can internally configure this, if needed.

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 with which to 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 explained above.
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 resource 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 are not distinct IP's then we will not race.
4. Edge servers ensure that a second request is not sent to the origin. This protects against request amplification.
5. The winner is determined by TTFB timing, and holds that IP 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 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..

Configuration Settings

The following configuration settings are available via the control panel.

Field	Default	Action
On/Off	Off	Enables and disables the feature, on or off.
URLs	Blank	Allows user to specify which URLs in their app actually trigger 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.

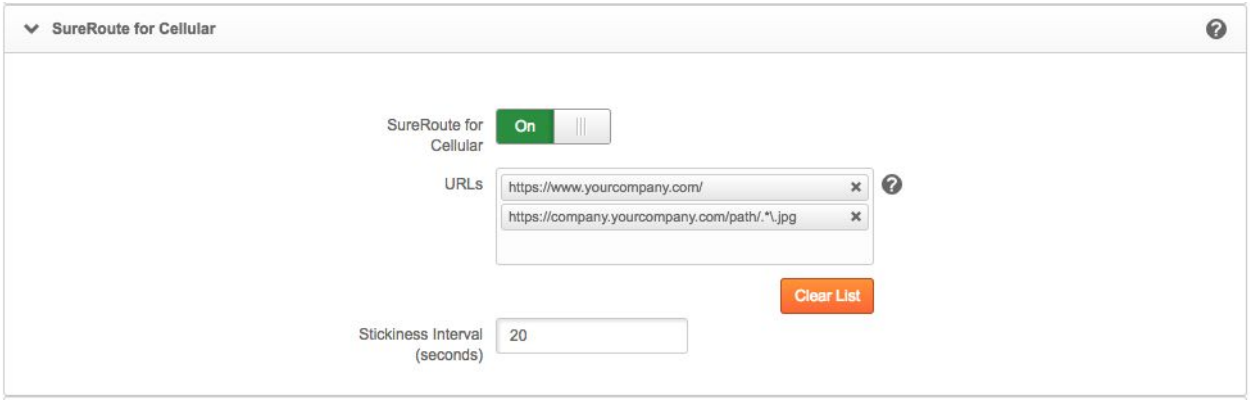
Usage

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 that are 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 Akamai 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 (max).

URLs

You can specify which URLs in your app actually trigger the second request by entering each one in the field provided. Note that you need to include the protocol and end with a backslash (see the following image). Regex expressions are allowed as wildcards.



Note that DNS lookups for different domains can resolve to different maps and IP addresses, so you don't want to run races on just the hostname. For example, if you have host.com with three subdomains, News (`news.host.com`), City (`city.host.com`), Hotel (`hotel.host.com`), you should 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 '?' then that must be escaped with a backslash ('\'). `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`

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 that's typically the best setting. However, if your main concern is uptime (you don't want to declare that all downloads use just the one winning path for all 20 seconds) you can set the interval setting lower. Setting the stickiness interval to zero seconds effectively runs a race one time and is used for a single download.

To illustrate this, let's take 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 20 seconds, then the same winning IP address will be used for up to 20 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 20 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.

Best Practices

- You should never use non-cacheable objects; it will cancel the secondary request.
- The more URLs you use, the better.
- Wildcards perform better than regex matches.
- The test object should be the first object requested.
- Have one test object per app view. For example, if you have an e-comm 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 use of QUIC (Adaptive Network Optimization).

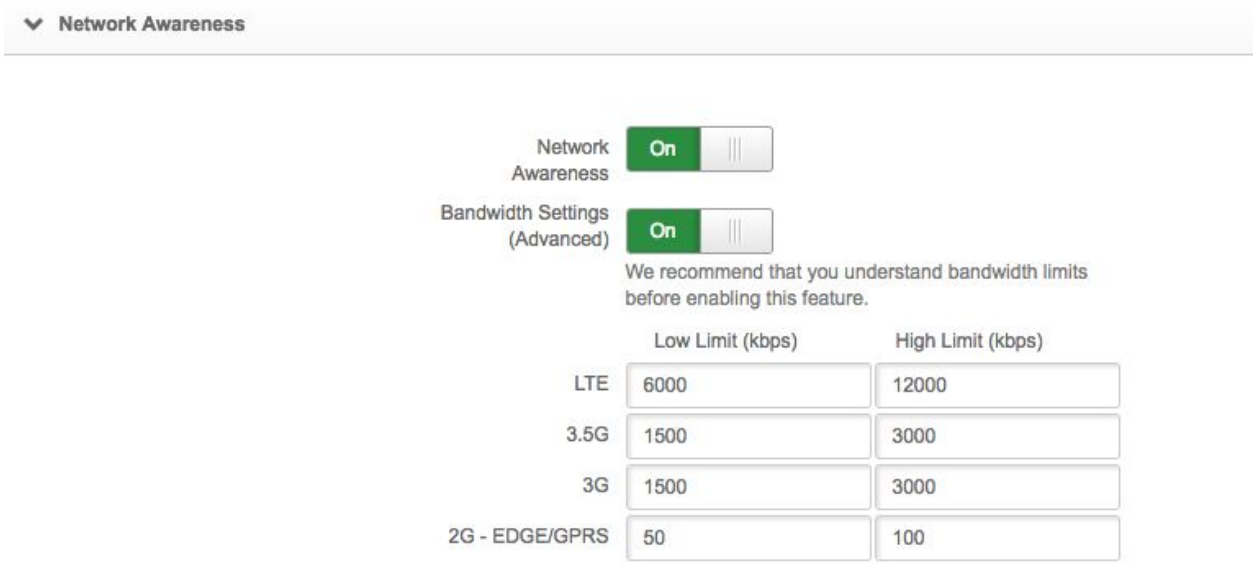
Network Awareness

Network Awareness monitors the network quality 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. We recommend using the default bandwidth settings, leaving Bandwidth Settings off.



Security Considerations

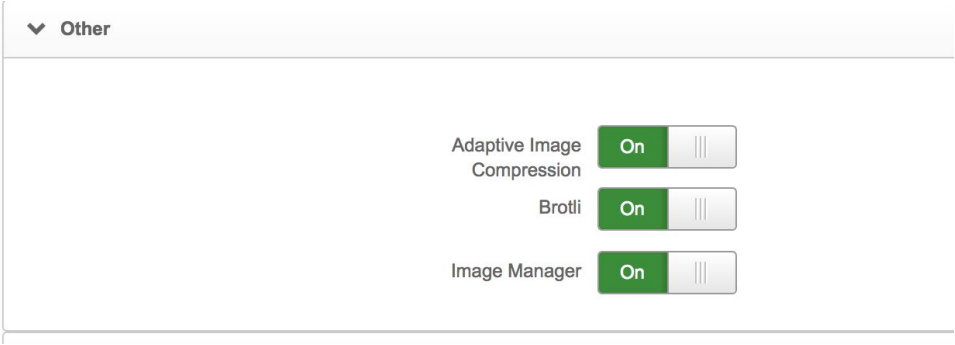
If you have Edge security features enabled, they may strip off the mobile connectivity header before it can be applied to image compression or reach the origin servers. Check with your PS consultant if you have Edge security features enabled.

Adaptive Image Compression

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 platform and the application running on the device, Akamai is able to make decisions to apply greater image compression to maintain a consistent user experience.

AIC is On by default. **To disable Adaptive Image Compression:**

1. Log into Luna and click **Configure > Mobile App Perf SDK**.
2. Scroll down to the Other category at the bottom and turn AIC to **Off**.



Note that Adaptive Image Compression is automatically optimized by the mobile-connectivity header values, while Image Manager is not.

Usage

Adaptive Image Compression (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

Adaptive Network Optimization (ANO) speeds up mobile apps by reducing time to first byte (TTFB), total download time (TDT), and packet loss, and 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 flavor 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.

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, with a high-quality carrier, an aggressive profile might start with a much wider ICW and react more slowly as the packet queue increases.

Adaptive Network Optimization is a key optimization capability of MAP SDK. It is configured automatically and adapts to changing conditions. Therefore, it is on by default, used for all communications between the device and the Edge (barring any A/B testing control groups), and 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 is integrated then the recommendation is to turn SureRoute for Cellular Off. This limitation does not apply to Android.

Brotli Compression

The Android Brotli library allows your Android app to accept BR 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:**

3. Log into Luna and click **Configure > Mobile App Perf SDK**.
4. Scroll down to the Other category at the bottom and turn Brotli to **Off**.

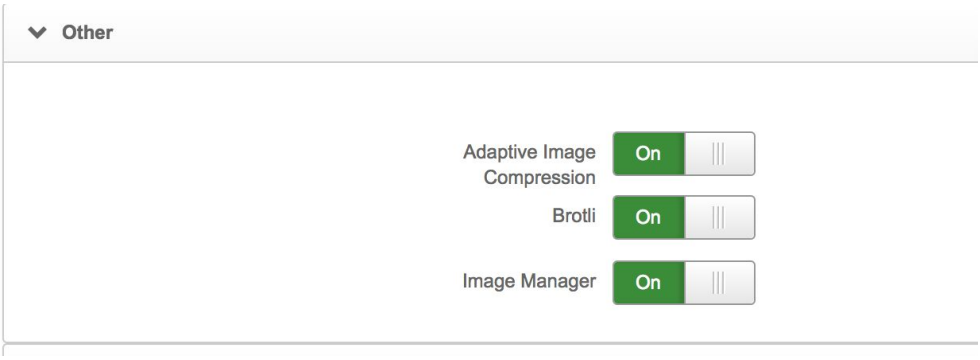


Image Manager

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 enabled by default, and you can disable it right next to Brotli, above.

To take advantage of this feature, you need Image Manager enabled and configured. When logged into the control panel, you can sign up for a trial in the Marketplace: https://control.akamai.com/marketplace/#!/products/image_manager/overview

Analytics and Reporting

The MAP SDK provides the capability to measure both standard performance events, such as time-to-first-byte and total download time, as well as custom events within the app.

Reports

The MAP SDK library also collects network-related statistics while serving content. These include HTTP time to first byte, 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 augment the user experience by taking necessary actions based on network state.

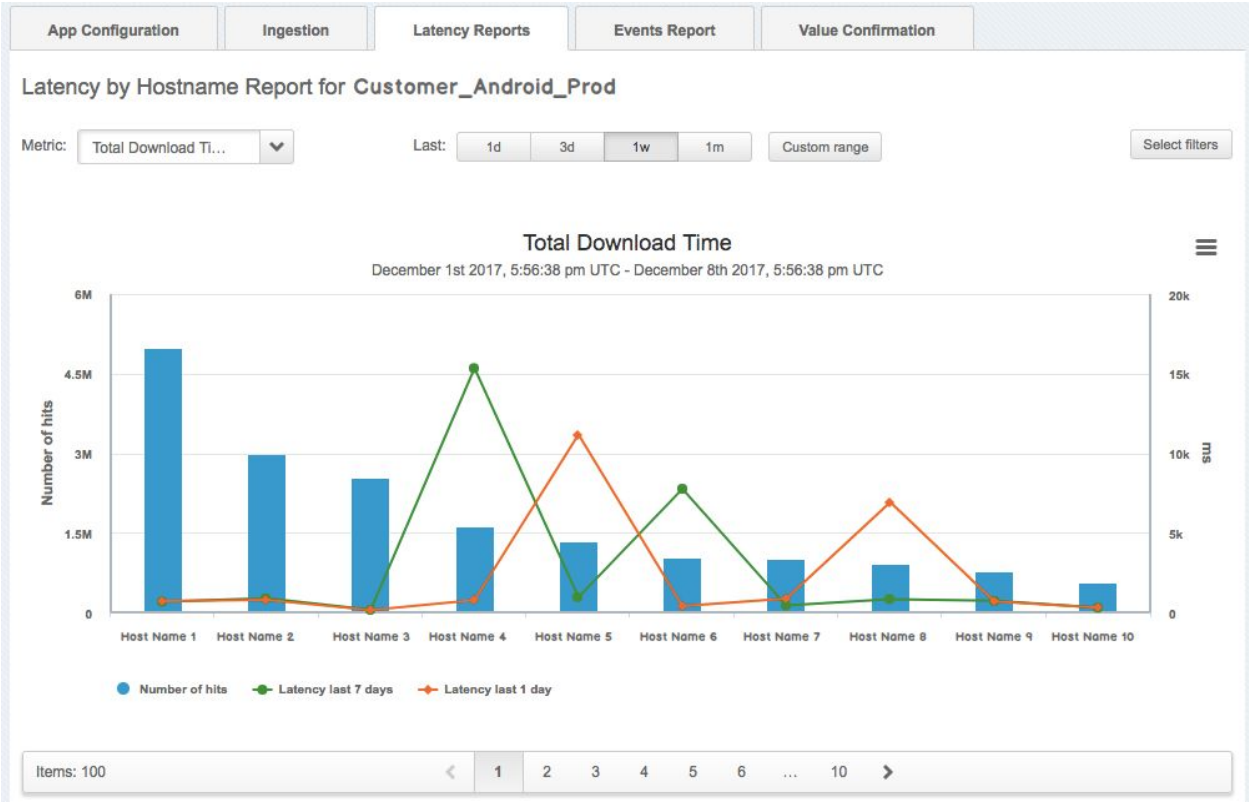
Latency Reports

The Latency Reports tab shows reports based on standard performance events, such as time-to-first-byte and total download time.

To view latency reports:

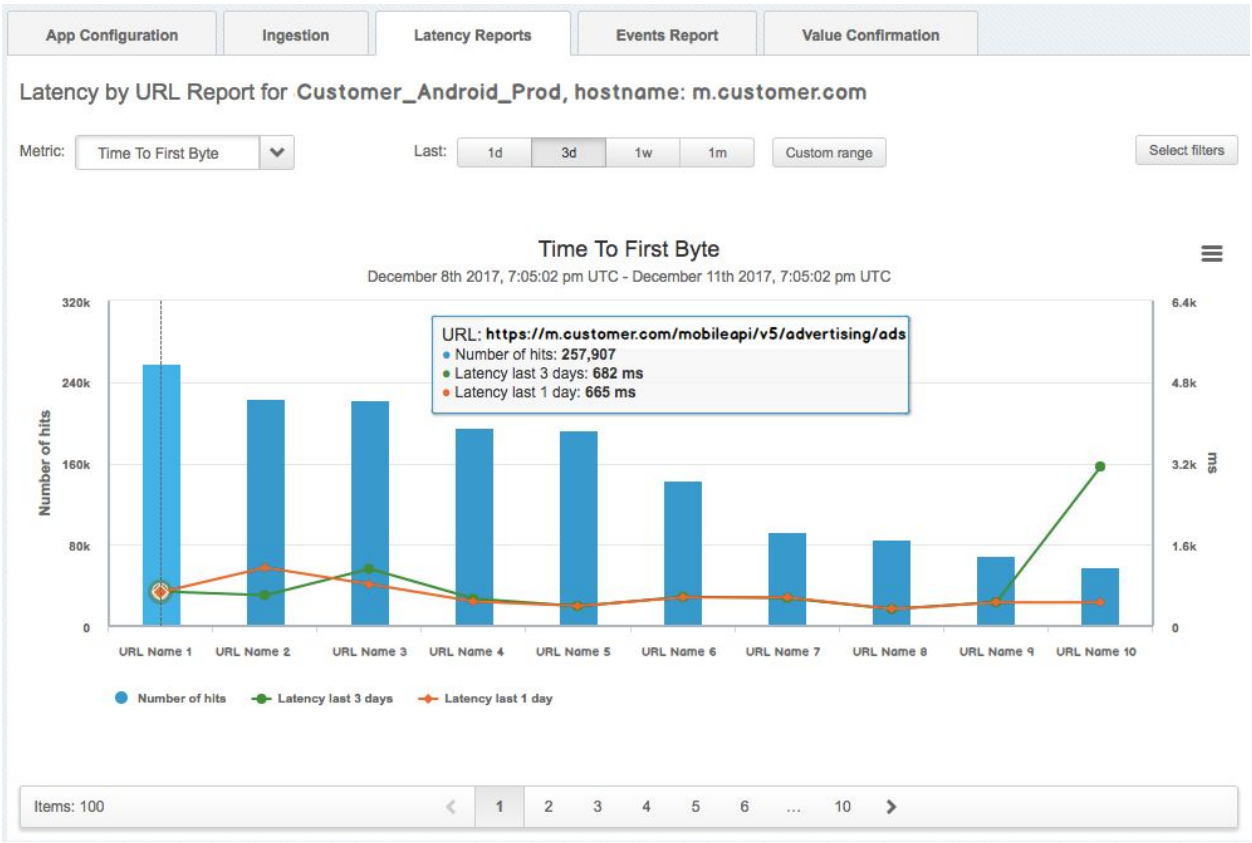
1. Log into Luna and click **Configure > Mobile App Perf SDK**.
2. Select a MAP SDK Policy, and then click the report icon (it looks like a bar graph).
3. Select Time to First Byte or Total Download Time, and whether to look at the last day, three days, week, or month or set a custom range.
4. Click **Select Filters** to further refine the report data.

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



View all URLs in a Hostname

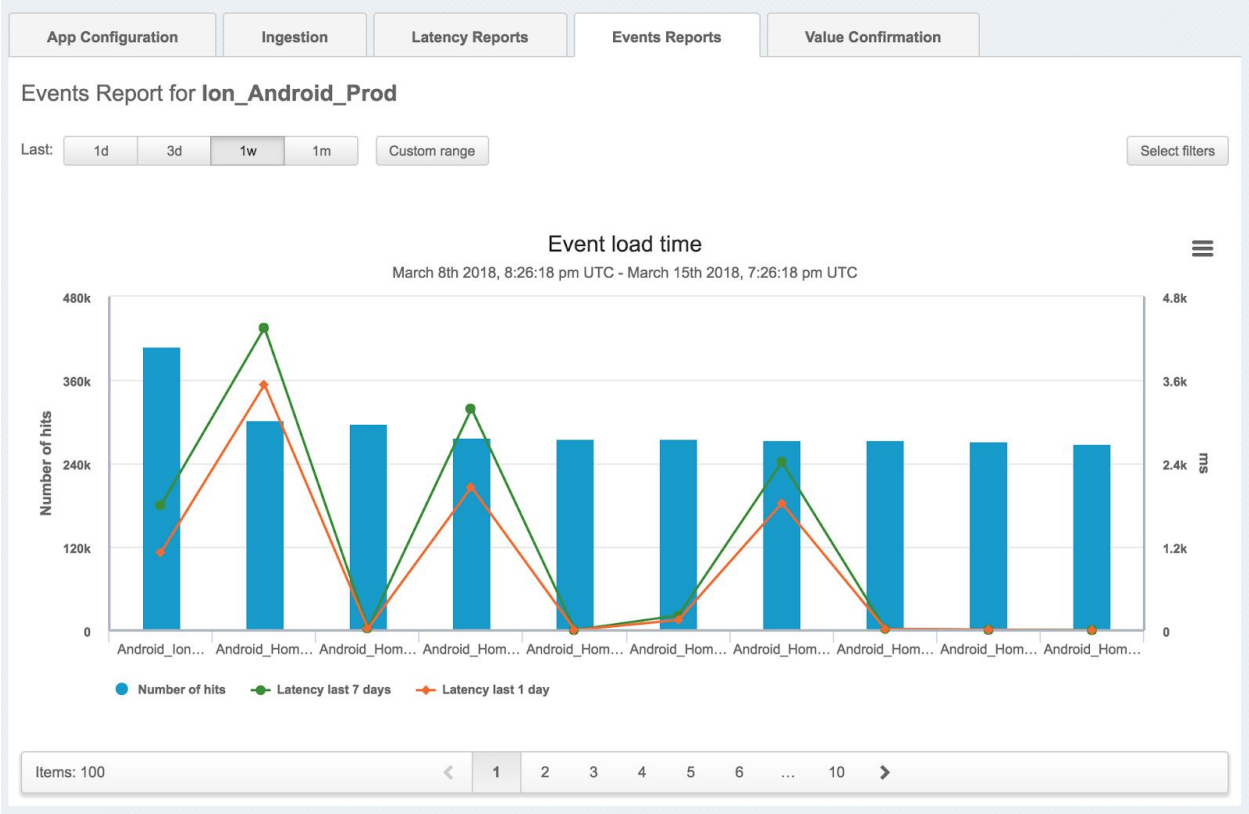
You can further drill into a hostname to inspect all of its URLs. Click one of the blue bars that represents a hostname and you can see the associated URLs, as in the following image:



Custom Event Reports

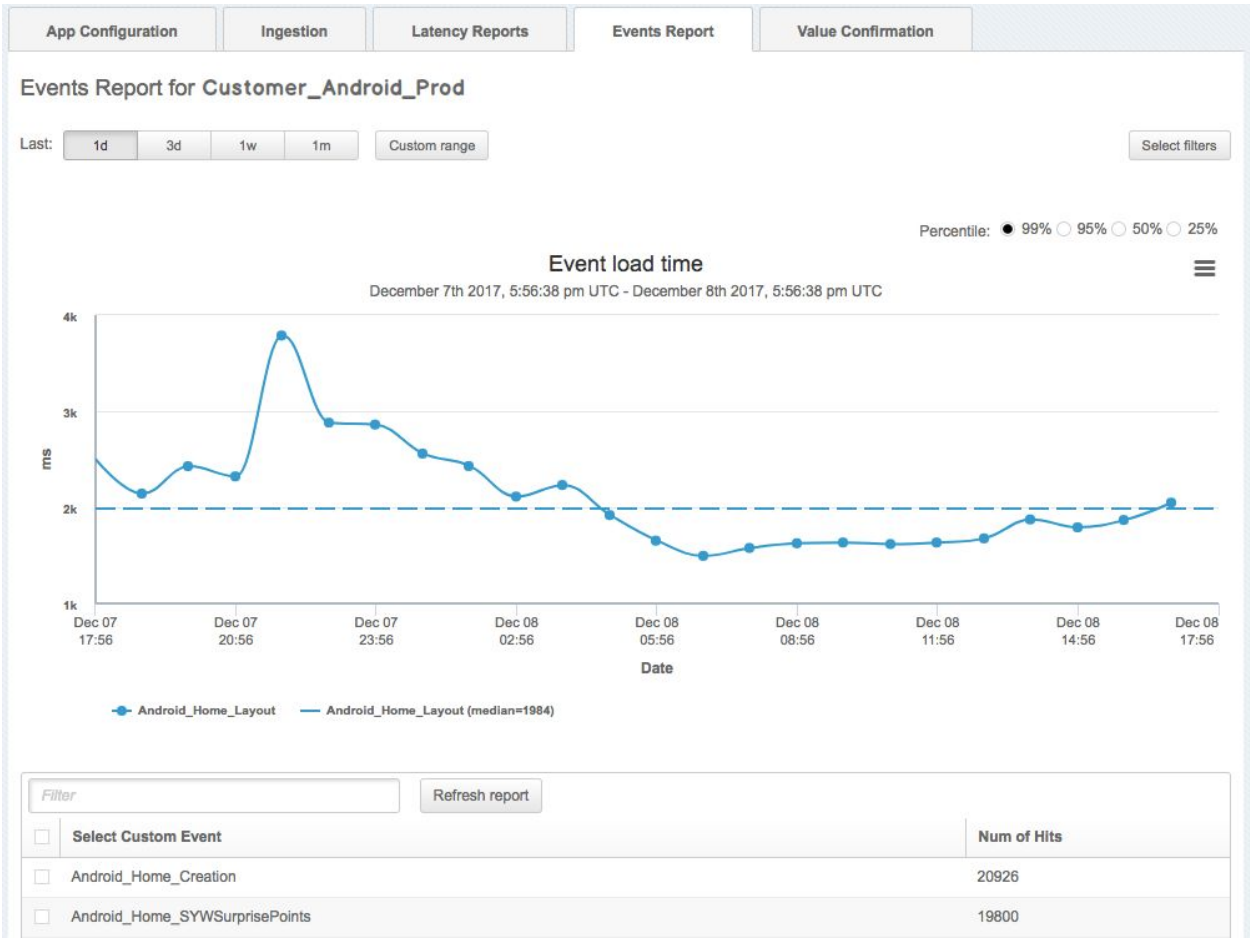
The Custom Events tab shows reports based on custom event timers that you've set up that measure how long it takes for a user to execute app-specific use cases, such as time to navigate to a product or check out. If you haven't set up custom events, you won't see any on this tab. Please refer to the Developer Guide in the SDK zip file in order to set up the custom events.

When you select Custom Events Reports, you'll see 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 drill down into each specific custom report, as shown in the following image:

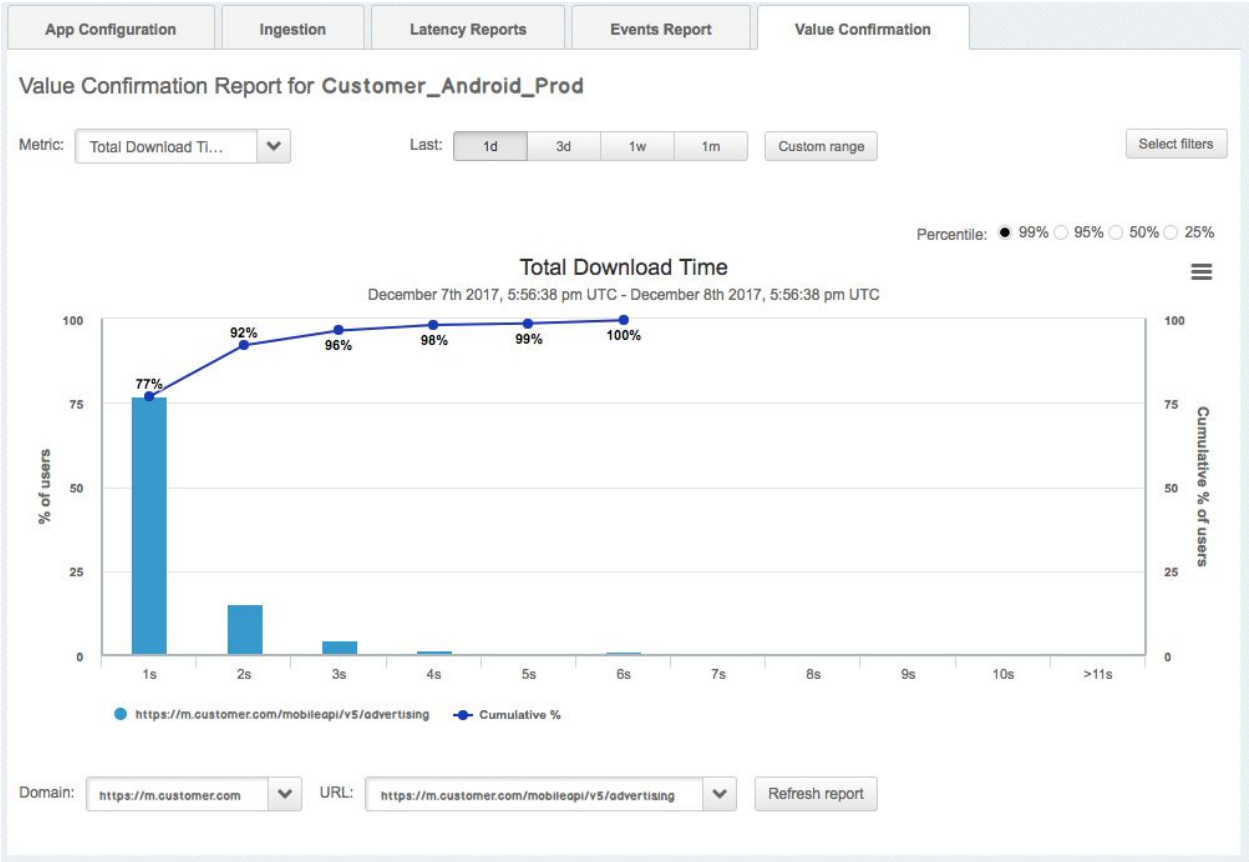


To view custom event reports:

1. Click the **Events Report** tab. By default the reports are displayed sorted by Number of Hits, but you can choose to sort alphabetically if you're looking for a report by its name.
2. Choose the Event Report you're interested in. You can also multi-select to compare multiple events at the same time.

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.



To view value confirmation report data:

1. Click the **Value Confirmation** tab.
2. Choose the **Metric** you're interested in (Time to First Byte, Total Download, etc) and **Select filters** appropriately.

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 certain percentage of users, you can:

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

You set the percentage use for each feature or universally, and the SDK automatically assigns the A and B groups. For example, let's say you're just starting to use SureRoute for Cellular and you want to see how a smaller test group compares to the existing app users. So you set the percentage field for S4RC to 20% On. The SDK then turns this feature on for 20% of users, creating an A group, and off for 80%, creating the B group. This allows you to test a features performance and reliability before rolling it out to all your app users.

Configuration Settings

The following configuration settings are 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), or on a per-feature basis (feature-level A/B testing). Universal A/B testing is on by default.
UC	Disabled	Sets the percentage for Universal Cache on a per-feature basis (feature-level A/B testing).
AIC	Disabled	Sets the percentage for Adaptive Image Compression on a per-feature basis (feature-level A/B testing).
S4RC	Disabled	Sets the percentage for SureRoute for Cellular on a per-feature basis (feature-level A/B testing).
Net Awar	Disabled	Sets the percentage for Network Awareness on a per-feature basis (feature-level A/B testing).
Brotli	Disabled	Sets the percentage for Brotli Compression for Android on a per-feature basis (feature-level A/B testing).
Image Manager	Disabled	Sets the percentage for Image Manager, if enabled and configured on a per-feature basis (feature-level A/B testing).
Pre-cache	Disabled	Sets the percentage for Pre-cache, if enabled and configured on a per-feature basis (feature-level A/B testing).

Device A/B Group Selection

The SDK determines groups A and B randomly for each feature on each device. Think of it as rolling a random number with dice. If multiple features are being A/B tested at the same time, then there's a roll for each feature. A device can be in group A for one feature and group B for another.

The A/B settings apply until the next configuration is received. After a new configuration is applied, the random test groups are re-rolled, even if the percentages are unchanged.

For example, imagine a device that has SureRoute enabled at 75%. The SDK rolls the dice and SureRoute goes in the A group (On). Later, a new configuration is received that adds Image Manager for A/B testing as well. The SDK rolls the dice for Image Manager and puts it in the A group (On) and re-rolls the dice for SureRoute (even though it was already enabled and the percentage was the same), this time putting this feature in the B group.

Feature-level A/B Testing

Features marked for A/B testing have a percentage field activated when the feature is On and Off. You can turn on all features at a certain percent, or toggle the percentage of each feature. Up to two features can be tested at a time.

Universal A/B Testing

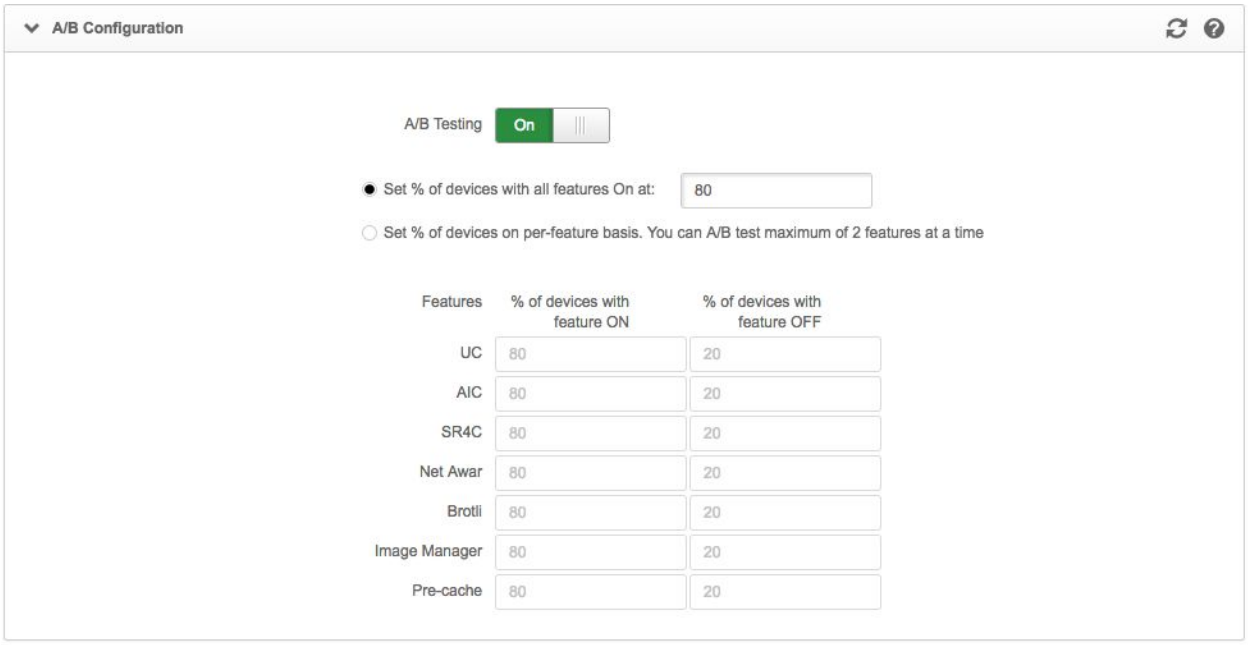
Universal A/B testing works a bit differently than feature-level testing, because the SDK rolls the dice once to determine if all features are in the A or B group, and applies that setting to all features that are enabled. When you do a universal A/B test, it overrides any individual A/B numbers, but doesn't activate features that are off; if a feature is not enabled, it remains Off.

Note that the universal A/B result is re-rolled whenever the device receives a new configuration, even if the percentages have not changed. Applying a new configuration effectively starts a new test.

We recommend setting the universal 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 users who have no features enabled, allowing a comparison against other users.

Universal A/B Example

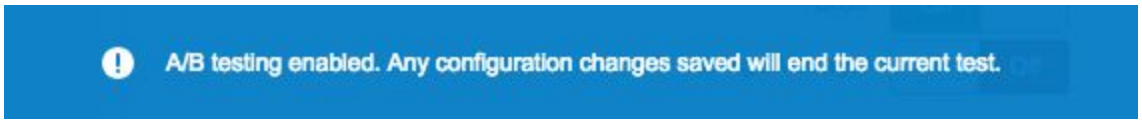
To better understand how universal A/B works, take the following example where universal A/B is enabled (first option button), with the A/B percentage set to 80% On. Here are the settings:



When the mobile app is launched (or after getting a push indicating a configuration change), the SDK rolls the dice and places the device in the A group 80% of the time, and into the B group 20% of the time. If the device goes in the A category (On), then AIC, S4RC, and Brotli are all on, regardless of how their individual percentages are set. However, the On/Off value is respected, so Universal Cache, Network Awareness, and Image Manager are not enabled. If the device goes in the B group, then no features are enabled.

Considerations

It's important to note that changes to configuration while A/B testing is enabled makes for confusing results, so as a reminder you'll see a warning when A/B testing is turned ON and when a change in configuration is made.



As the global leader in Content Delivery Network (CDN) services, Akamai makes the Internet fast, reliable, and secure for its customers. The company's advanced web performance, mobile performance, cloud security, and media delivery solutions are revolutionizing how businesses optimize consumer, enterprise, and entertainment experiences for any device, anywhere. To learn how Akamai solutions and its team of Internet experts are helping businesses move faster forward, please visit www.akamai.com or blogs.akamai.com, and follow @Akamai on Twitter.

Akamai is headquartered in Cambridge, Massachusetts in the United States with operations in more than 57 offices around the world. Our services and renowned customer care are designed to enable businesses to provide an unparalleled Internet experience for their customers worldwide. Addresses, phone numbers, and contact information for all locations are listed on www.akamai.com/locations.