APPTIMIZE

QA Guidelines

Overview

Like any other update you make to your app, it is very important to also test the experiments and features that you deliver to your end-users using Apptimize. This document highlights some of the common approaches that our customers take to QA their projects in Apptimize.

The 3 main QA approaches described in this document are:

1. Using Preview Variants
2. Using programmatic APIs to force variants
3. Using selective targeting to launch projects internally within the team

We recommend using these tools and resources to test the behavior of any project before launching it to the end-users. Some of the common things to look for are:

1. Verify that the **participation for the experiment is recorded** where you expect the user to be exposed to the experiment
2. Verify that the **events are getting picked** up by the Apptimize SDK
3. Verify that the **participation gets recorded before the events** relevant to the experiment
4. Verify that the **metadata is received in time** to deliver the experiment
5. Verify that visual edits display properly across a variety of devices and resolutions

General best practices

- As a general best practice, we always recommend that customers create separate apps in Apptimize for their **Staging and Production environments**. This allows them to freely and extensively test out experiments and feature flags without worrying about any adverse effects on the end-user experience. This also keeps the production dashboard clean and manageable, and prevent results generated during testing sessions from polluting results from end-users

- For teams that have the necessary development resources or automation processes, we highly **recommend always using the programmatic approach** for QA along with using Preview Variants to test the variants.

- It is recommended to make use of the **tags and notes** fields as much as possible and to integrate updating these fields into the day-to-day experimentation process. These fields can be used to store Jira ticket numbers and links to specs so that it is easier to track and reference the stories related to the experiment or feature flag.

- It is recommended to use **multiple devices** to test out an experiment before releasing it to the end-users, especially for visual experiments. It is a good idea to use devices with different screen sizes to verify that the changes appear as expected on all screen sizes.

- It is crucial to verify that **participation events are reported where expected** in the app and that any events that are important to measure the success of the experiment are recorded after the participation event. Apptimize results only show conversions for events that occured after the user has participated in the experiment.

- Ideally, you should always **do all your QA in your Staging environment** but if you were to do it in the production dashboard in Apptimize, remember to create a fresh copy of the experiment and archive the original one so that any results introduced during QA are not a part of the actual experiment results.

Preview Variants

Preview Variants is a tool that allows you to pair your device to the Apptimize dashboard and test the variant on your actual device so you can verify that the variants are performing as you expect them to.

When connected to Preview Variants, it provides a visual representation of the variant on the Apptimize dashboard via mirroring, a participation indicator, a section to verify the targeting and representation of the captured events. We recommend this approach of testing for simple visual/UI debugging (especially if the Visual Editor was used to create variants) and/or event-tracking.

Advantages

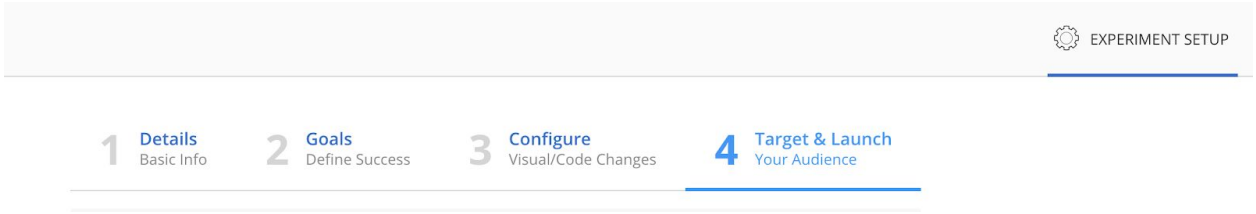
- Zero coding involved
- Works hand-in-hand with previewing changes made in the Visual Editor

Limitations

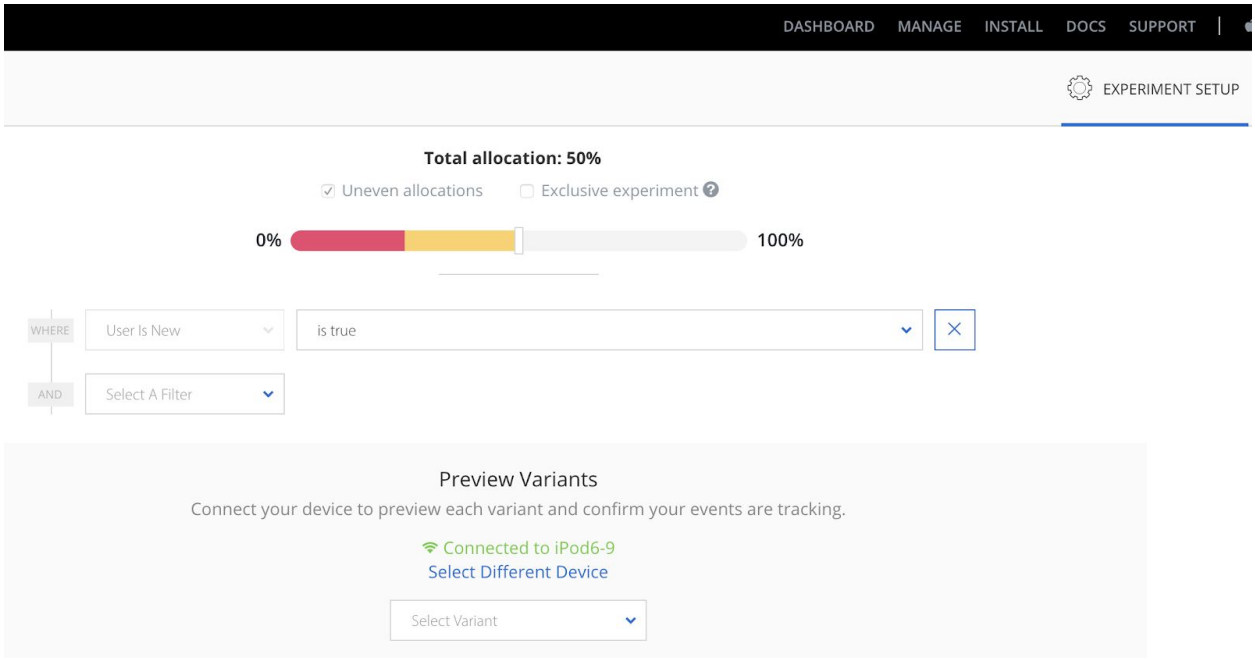
- You can only preview one variant at a time. (i.e. while using this tool, all your other active experiments and feature flags will be disabled on your device)
- The variant is applied to the device as an Instant Update instead of an experiment. Hence, APIs like testInfo, getVariants or onExperimentRunListener do not work with Preview Variants.

How to Pair Your Device

The Preview Variants section can be found on Step 4, **Target & Launch**, of the Experiment set-up process:



Assuming you've configured your visual/code changes for variants (from Step 3), you can then scroll down to Preview Variants from the Target & Launch page and find any eligible devices for pairing.



If you are using a **development build** of your app, you should see your device available for Preview Variants as soon as you open your app. If you do not see any devices available, please check the following:

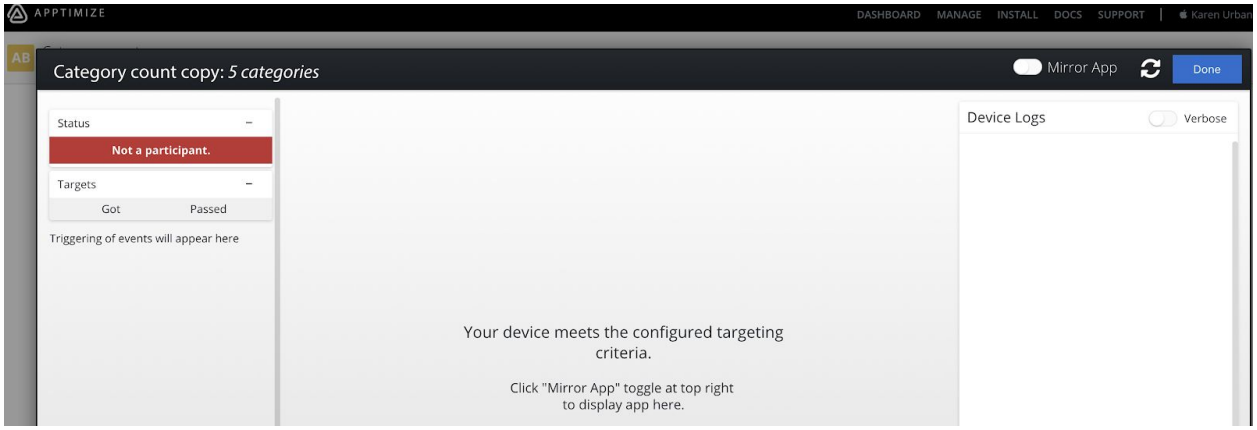
- Ensure that you've followed the steps on the Install page and verify that the Apptimize SDK is properly installed
- Make sure that your unique application key (also found on the Install page of your experiment) is implemented in your app
- Make sure your application is open and running
- For additional details see this Support FAQ

If you would like to test your app using a **production build or a signed build**, you will need a unique pairing token. You can obtain the pairing token by emailing support@apptimize.com or your CSM.

Once you have your pairing token, you can just copy the token to the clipboard before launching the app (remember to kill and relaunch if app is already open) to pair with the dashboard.

The Preview Variants Modal

Once you're able to select a device, choose a variant to preview from the dropdown. This will launch a floating window similar to below:



- The experiment name and variant name are displayed in the top left
- Toggle "**Mirror App**" to mirror the display of your currently paired device
- Click the "**Refresh**" icon to refresh the display
- On the left-hand side, the participant's "**Status**" will be displayed.
    - For visual experiments the status will change to "Participating" if you are on the view that includes the variant, for code block experiments participation is triggered when the code block is executed, and for Dynamic Variable experiments it occurs when a dynamic variable involved in the experiment has its value queried. This is confirmation that participation is recorded.
- The **"Targets"** section will display the targeting for the experiment
    - If you have any targeting criteria, the values (for example, App Version: 2.0) as well as an indication of whether the current device meets it will be displayed here. Participation cannot be triggered unless your device meets all of the experiments targeting criteria.
- An **"Events"** section will appear once you trigger an event in your app
    - Any triggered events will appear here with the associated value (if available) will be displayed in this section
    - Any events with a "to be imported" status signify that the events are successfully detected and will be imported approximately one hour after the event is triggered on a device which is enrolled in a live experiment.

What are other customers doing?

Preview Variants is a useful tool for front-end/UI testing. Product managers often prefer to do preliminary testing with the tool to confirm variants/events are appearing as they should - this is straightforward and fast due to the lack of coding required.
For complex QA testing beyond visual elements, our clients prefer to use our programmatic or targeting-based QA. These methods are outlined in the remainder of this document.

Programmatic APIs for QA

Advantages

- Variants from multiple experiments can be forced simultaneously
- The (forced) state of the variant(s) persists across app kills and relaunches
- Data exporting APIs are available to be used (unlike Preview Variants)
- Ideal for functional testing as well as unit testing

Limitations

- Requires development
- The experiment needs to be running

Useful APIs

Programmatically force variants

The forceVariant APIs provide you with a programmatic way to force certain variants to your device using the variant id.

Once forceVariant is called, Apptimize is placed in a special test mode where it will only enable variants that are forced by forceVariant. All other Feature Flags, A/B Experiments and Instant Updates will appear disabled/off unless a specific variant is forced for those projects. You need to call forceVariant for each of the variants you want to apply. So, if you have 3 experiments and you want to see how the variants from all 3 interact with each other, you must call the forceVariant API for one of the variants from all 3 experiments.

Note: Preview Variant will not work when there are forced variant(s) on a paired device. This is because Preview Variant works by sending an instant update to the paired device.

iOS APIs

(void)forceVariant:(NSInteger)variantID
Force the variant with the given ID to be enabled. Filter logic is ignored. No other experiments or instant updates through regular means will be applied if a variant is forced.

If called multiple times, the set of variantIDs that are set via forceVariant are all forced. If an ID is invalid it does nothing, but other experiments / instant updates are still off.

(void)clearForcedVariant:(NSInteger)variantID
Disables the variant with the given ID. Does nothing if the variant is not forced.

(void)clearAllForcedVariants
Disables all variants which were forced by the forceVariant call.

Android APIs

public static void forceVariant(long variantID)
Force the variant with the given ID to be enabled. Filter logic is ignored. No other experiments or instant updates through regular means will be applied if a variant is forced.

If called multiple times, the set of variantIDs that are set via forceVariant are all forced. If an ID is invalid it does nothing, but other experiments / instant updates are still off.

public static void clearForcedVariant(long variantID)
Disables the variant with the given ID. Does nothing if the variant is not forced.

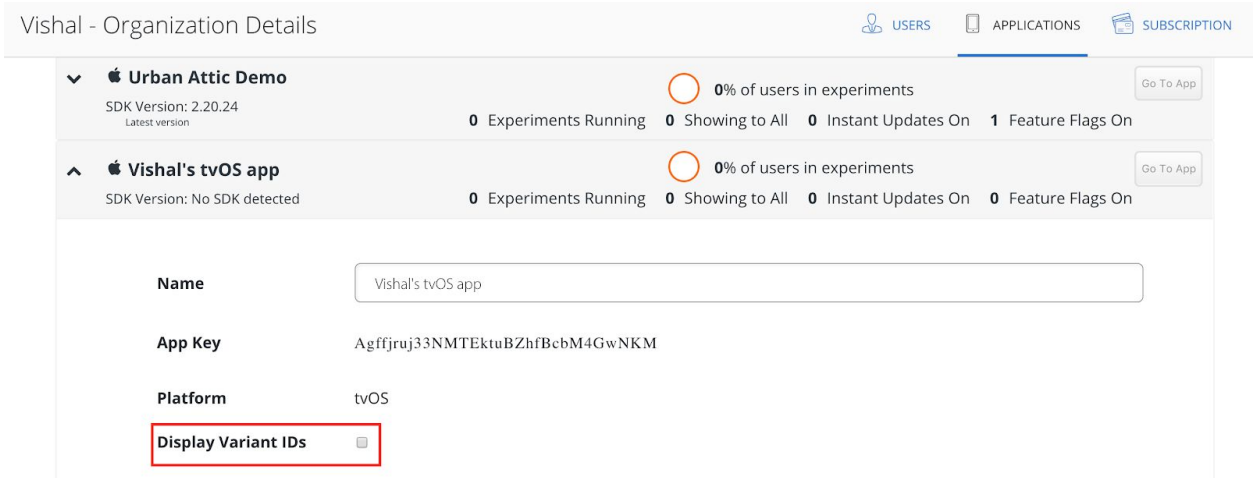
public static void clearAllForcedVariants()
Disables all variants which were forced by the forceVariant call.

Getting the variant ids

There are 2 ways to get the variant ID

1. You can use the getVariants API (iOS and Android) to retrieve the information about all the variants of the currently active projects.
2. Alternatively, you can view the id for a specific variant in the experiment configuration in the configure step or in the results, next to the variant name.

Note that if you do not see the variant ids in the dashboard, you can enable the feature in your 'Organization Details' view, by checking the 'Display Variant Id' option in the detail view of the app.



Managing the verbosity of logs

You have the option of controlling the verbosity of Apptimize logs, using the Log Level option. This option can be configured during the Apptimize setup call to get detailed logs that can be useful for troubleshooting.

iOS API
Android API

Note that in iOS, this option can also be set in the plist using the property ApptimizeLogLevel.

What other customers are doing?

Some of our customers have implemented a QA console by binding the above API calls with some basic UI (toggles, buttons for clearing variants, etc). In the testing phase they append their developer build with this console to allow for easy testing without having to constantly amend the application code.
We currently have a work-in-progress build of a console that accomplishes some basic forceVariant functionality. Note that this is a feature we are presently refining based on QA needs.

Targeting based QA

Sometimes the best way to QA is to experience a variant exactly like an end-user would experience it. That way you could verify experiencial things that might otherwise get missed in regular testing. This is also good to check that the experiment does not introduce any lag in the app or that the results are being collected and displayed as expected.

This approach specifically works well for customers with larger QA teams where it might be challenging to distribute development builds or teach everyone how to use pairing tokens. This can be achieved by using some creative targeting criteria. Using this approach in conjunction with verbose logging makes it very simple to troubleshoot experiments and feature flags.

Advantages

- Easy set up
- Closest experience to end-users
- Allows QAing multiple experiments simultaneously
- Results will be reflected in the Apptimize dashboard

Limitations

- Can only troubleshoot with logs
- Since the assignment to the test happens in a way similar to how it will happen once the experiment is launched to your end-users, it is not possible to control which variant to push to device. However, you could select a winning variant if you want to test a specific variant.

Examples

Use Custom Attributes for targeting

You can set any user characteristic that is available in your app as a Custom Attribute to target your user cohorts.

For example: You can use the email used to sign in, to set a Custom Attribute that tells you if a user is an employee of the company and target the experiment to be delivered to them.

Link to Documentation: Custom Attributes

Use Pilot Groups for targeting

One of the most common ways customers use to target experiments to the internal team is using Pilot Groups. Groups can be created in Apptimize based on the values of a pilot targeting id that can be set in your app. The pilot targeting id can be any attribute in your app and can be

set up very easily. Once the pilot targeting id is set in the code, we recommend creating a group for your QA team by uploading their ids to the dashboard. This way every time you want to send an experiment to only your internal team, you can easily do so by just adding the group to the targeting criteria.

Note that pilot groups are independent of targeting. Pilot users will always see your experiment or feature flag, regardless of whether they meet other specified targeting or version criteria.

Link to Product Documentation: Pilot Groups
Link to SDK Documentation: setPilotTargetingID function (iOS / Android).

Use version numbers for targeting

One of the easiest ways to target only the QA team for an experiment even if you do not have custom attribute or pilot targeting id set up in your code is to target based on app version. Usually the app version in QA is higher than that in the AppStore or PlayStore. So you can target the QA team for an experiment just by simply setting the targeting for app version to be greater than the current version of the app in the AppStore/PlayStore. Keep in mind that if you use this method, you must make sure to update the targeting criteria if an update to you app is released.