Quality assurance, or QA, is the last step in creating an experiment. A rigorous QA process ensures that your experiment looks and works the way you want before you show it to visitors. Once you’ve finished building your experiment, take a little extra time to verify your setup so you can trust your results.
To learn more, check out our [Optimizely Academy course, How to QA Your Experiment](#). Alternatively, scroll down to check out our QA tools.

## Materials and resources to prepare

Here is a recommended list of what you'll need before getting started with QA. Start with a QA Checklist. If you’ve followed the Optimizely documentation from the beginning, you likely have a [basic experiment plan](#). Use it as a template to outline the QA checklist. Major topics the checklist should cover are:

- All new functionality to check
- All goals and analytics to be captured
- All user flows into the experiment
- Verify that the new experiment doesn’t break existing functionality

Next, identify who will be involved with the QA process. Different types of experiments will require different types of people and resources. An experiment with custom code may require developer resources, whereas changes made with just the Visual Editor may not.

Consider the following list of people types:

- Developer or technical skills
- Experiment implementer
- Power user
- Optimizely practitioner
- QA team

When setting your experiment deadlines, give yourself enough time to go through the QA process fully before launch. Catching issues early helps avoid broken experiments, skewed data and unintentional impact on visitors.

- Plan time for the QA process
- Sync with QA team to understand the time required
- Allocate time and resources accordingly if issues with the experiment are discovered

Ultimately, the most effective QA process really depends on the processes that are in place in your organization. Use the framework in this article to build rigor into your QA steps and integrate your testing process with company practices.
Environments

Many mature optimization teams have access to separate staging and production environments. Separate environments can save you a headache by reducing the risk of accidentally launching an unfinished or unverified experiment to the live site.

If you have separate environments, use a staging environment to set up the experiment and perform initial QA without exposing the experiment to visitors. Then replicate the experiment into your production environment project, and perform the same QA process there. When possible, ensure that your production and staging environments match so experiments function the same way in both.

If you are using environments, consider the following steps when building a QA checklist:

• Verify that components in the production environment match the staging environment
• Verify that the experiment functions correctly in staging
• Verify all experiment changes do not impact site functionality
• Run QA steps in the staging environment
• Run QA steps in the production environment

When this feature is ready, make sure this is included as a Note:

The environments feature allows experiments to be built in one project, but allocated to either a development or production environment. There is no duplication or rebuilding required. See our Knowledge Base article for more details.

If you don’t have a dedicated staging environment, don’t worry. You can still QA your experiment. Follow the steps outlined below, but take special care when you use the test cookie to QA in a live environment.
Basic QA steps before launch

When you QA, you'll test three functional areas for every experiment: variation code, targeting and activation, and metrics. This section describes the QA needs of each of these, and outlines some key questions to ask during the QA process. These steps all require the use of the Optimizely Preview Tool.

Variation code

Does your variation look the way it should? How your experiment is presented will almost certainly have an impact on the results, so make sure you've got it the way you want it.

As you built your experiment, you may have noticed the Visual Editor actively responding to your changes. You can use Optimizely's Preview tool to run the changes on the webpage outside the Visual Editor environment.

When building your QA checklist, consider the following points:

- Do the changes you made appear on the page?
- Are the changes appearing where expected?
- Does dynamic content still work as expected?
- Does the variation work in desktop, mobile and tablet views?
- Do the changes work across different browsers?
- Do you observe a flash?

If you're having problems with this, see our article on troubleshooting variation content.

Targeting and activation

Targeting defines where the experiment will run. To check that the experiment will activate on the appropriate URLs, open the preview tool to the Currently Viewing tab. If your experiment is activated, you should see the name of the campaign, the name of the experiment, and the name of the active page listed there.

If you are running a multi-page experiment, navigate to each URL and ensure that the page activates in the preview tool. The preview tool is persistent, so navigate to and from URLs included in your experiment's targeting to ensure that the experiment activates and deactivates as expected.

Checklist additions for targeting should include:

- Does each URL included in targeting activate the experiment?
- Do URLs not included in targeting activate the experiment?
- If you are using conditional activation, does the Page activate 'when' expected?
• If using support for dynamic websites, are pages deactivated when conditions are false?

• Are experiment changes removed when the page is deactivated?

If you’re having problems with this, see our Experiment is not activating article.

**Metrics**

Before your experiment runs, you can use the preview tool to confirm your events are firing as expected.

The event feed references the "saved" instance of Optimizely, and not the snippet that is published on the CDN.

Events that aren’t published or attached to the experiment can be triggered in the feed. Keep in mind that events triggered during preview are not recorded on the Results page, even if the experiment is live.

Event checklist additions:

• Does the click event attached to an element convert when clicked on?

• If multiple elements are targeted by a single click event does each one trigger a conversion?

• Are events triggered on elements that appear after a visitor takes a certain action?

• Are events triggered on elements that appear late on the page like a modal or pop-up?

• If you are working with a redirect experiment, have you set up a hybrid page to track events on both the original and redirected URLs?

• If using custom events, is the code calling the API working as expected

• For events attached to form submissions, are events fired when a visitor clicks the button or hits the 'return' key

• If there are errors from an incomplete form, should the event fire?

If you’re having problems with this, see our article on what to do when metrics don’t track correctly.

An easy way to share your variation designs with stakeholders is to send a share link. However, visitors who enter through the force variation will only see the variation you selected on the specified page; they won’t be able to navigate through multi-page experiments. If you’re making changes to multiple pages, use the appropriate link or QR code of that specific page.

---

**Advanced QA steps**

Once you experiment passes these basic QA checks, we recommend doing a final round of QA with the experiment running live. If you run the experiment in either a development environment (with no live traffic) or in a production environment with a test cookie approach, you can prevent visitors from inadvertently seeing your experiment before
you're satisfied it's ready.

Why bother running an experiment live at all? For one thing, it's an essential step for testing audience conditions, which the preview tool doesn't address. The test cookie approach gives you the power to view the experiment as your visitors would, trigger events that will show up on the results page, and expose issues (like, for example, timing) that might otherwise slip under the radar.

If your company doesn't allow adding cookies to staging or production environments, you can still create heavy audience restrictions that will prevent you from showing the experiment to visitors while working through the QA process.

When doing QA on a live running experiment, it is advised to use a new incognito window for each evaluation. Optimizely tracks visitors by storing data in the browsers local storage. To ensure that there is no previous session data, and no prior bucketing influencing your QA evaluation of the experiment, use a new incognito window to give you a clean slate.

Advanced event check

There may be times where you want to confirm both that an event is triggered, and that the Results page is receiving it. In the case of revenue or non-binary events, it may be important to confirm the correct value is passed in the API call. For these situations when the experiment is running, you can check if an event is firing live. Use the network events to check the values sent to Optimizely as they happen.

- Open a new incognito or private browsing window, and go to the page(s) you want to test and set a test cookie
- Go to the Network tab of the developer console
- Perform the action that you expect to fire the goal in Optimizely. Look for the event corresponding to that goal in the Network tab to see if it appears when you expect it to
- Click all the element/area that should trigger a click events
- Navigate to URLs tracking pageview events
- If the action involves moving from one page to another, check the Preserve log box in the network tab to track the network call across pages
- If triggering non-binary events, check that the event has the correct value passed in the network
- On revenue specific metrics if multiple currencies are accepted are they converted correctly
- Check your Results page and your analytics integration to ensure that data is captured correctly

Activation and bucketing issues

Sometimes the QA process will turn up a component of the experiment that's not working as expected. It's not always easy to immediately understand what the root cause of the issue is. In these cases, the JavaScript API and
the Optimizely Log can be extremely helpful in pinpointing where and why an issue exists.

Instances where the log and JavaScript API can lend themselves to resolving issues:

- If an experiment is part of an exclusion group are visitors excluded as expected
- Identifying if a visitor is in the holdback or not
- Explaining why an experiment is not activating when expected
- Looking at if an audience condition is succeeding or failing for a specific experiment
- Manually activating pages or manually sending events

Analytics and other third-party platforms

The best time to confirm that analytics platforms like Google and Adobe are receiving data with the correct experiment and variation information is when the experiment is running live. Optimizely integration logic is not run in preview mode; instead, Optimizely will only evaluate and pass the information on to analytics platforms if the experiment is in an active state.

For the checklist the following items should be included:

- Expected experiment and variation IDs are passed to analytics
- Analytics network event contains the experiment and variation information
- Analytics integration captures data within a custom report
- Analytics integration data aligns with what is expected
- Failure in audience or URL targeting is reflected in the integration setup
- If traffic allocation is less than 100%, is the holdback passed to analytics as expected

Deliverables

Once you have finished your QA, you should have the following:

- A fully built experiment in a production environment
- A completed QA and use case doc, with all cases marked "pass"
QA tools and resources

- **The Optimizely Chrome extension** is a must have if you do QA regularly. This tool gives you a quick read of active pages, as well as of your bucketing for active experiment and variation. The information this extension provides can substantially shorten your time spent on QA.

- The **preview tool** is your first line of defense in Optimizely X. Use it to check visual changes and functionality without publishing your experiment or campaign.

  The preview tool allows you to view all the experiments and campaigns on any page on your site, whether it's unpublished or live to visitors. You can check how your variations and experiences appear to different audiences, and verify that events are firing correctly.

- **The share link feature** helps you share specific variations with internal stakeholders.

- Use the **cross-browser test** feature to see how your visual changes look in different browsers and on different devices.

- The **test cookie** feature also helps you QA a running experiment and share it with internal stakeholders, without exposing it to your visitors.

- Use the **force variation parameter**. Data from draft and paused experiments are excluded from Optimizely X by default, so a force parameter like \( \text{?optimizely_x=VARIATIONID} \) only shows data from live experiments.

  To preview draft or paused experiments, you can add \( \text{&optimizely_token=PUBLIC} \) to the force variation parameter above, or use the Share Link or the Preview Tool.

- **The JavaScript API** provides an easy way to check what live experiments and campaigns are running on a page and into which variation you're bucketed.

- The **network console** helps you verify whether events in a live experiment or campaign are firing correctly. Use it to check that metrics are tracked correctly on your Results page.

- **The Optimizely log** helps you diagnose more difficult issues in a live experiment or campaign. It tells you whether an experiment or campaign activates on the page, whether you qualify for an audience condition, and whether changes on a page are applied.

---

**Example QA checklist**

The [QA checklist](#) that you created as part of your experiment plan will help you perform a thorough check.

As a reminder, your checklist should include:

- All goals that were added, as well as how each is triggered
- All functionality that’s been added (for example, a new button)
- Every visitor use case, including all expected user flows to and from the page
- Audiences that are eligible and ineligible to see the experiment
- URLs where the experiment should and should not run
- Sample workflow to fire a goal (especially for custom events)

<table>
<thead>
<tr>
<th>Audience visitor path</th>
<th>Eligible for the experiment?</th>
</tr>
</thead>
<tbody>
<tr>
<td>A visitor from Bing</td>
<td>Eligible for experiment</td>
</tr>
<tr>
<td>A visitor from Yahoo!</td>
<td>Eligible for experiment</td>
</tr>
<tr>
<td>A visitor from Google</td>
<td>Eligible for experiment</td>
</tr>
<tr>
<td>A visitor who clicks a paid ad</td>
<td>Not eligible for experiment</td>
</tr>
<tr>
<td>A visitor who clicks an email link</td>
<td>Not eligible for experiment</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Experiment Metrics</th>
<th>Location / Behavior to Trigger?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Click on CTA</td>
<td>Click on the 'Learn More' Button in hero image on example.com</td>
</tr>
<tr>
<td>View Checkout Page</td>
<td>navigate to URL example.com/checkout</td>
</tr>
<tr>
<td>View 3 Card Promotion</td>
<td>Scroll 60% of the way down the page on URL example.com/promotions</td>
</tr>
<tr>
<td>Track Revenue</td>
<td>On all Order Confirmation Pages confirm value sent is correct</td>
</tr>
</tbody>
</table>
Once your QA team verifies every item on your QA checklist, you're ready to launch the test to visitors. Remove the test cookie audience condition, and then click the Publish button.

Congratulations! Your experiment is now live for visitors to your site.