THE FEEDBACK PLAYBOOK

EVERYTHING YOU NEED TO KNOW ABOUT CULTIVATING
HIGH-QUALITY FEEDBACK DURING BETA TESTING



centercode

TABLE OF CONTENTS

INTRODUCTION 1

- **The Value of Quality Feedback** 2
- **The Beta Testing Toolbox** 3
  - Ongoing and Directed Feedback 3
- **Feedback Collection Psychology** 4
  - Maximizing Energy Pools and Reducing Friction 5
  - Validating Your Beta Testers 6
  - Setting Participation Expectations 7
  - Collecting a Variety of Feedback 7
  - Balancing Testers' Activity 8
  - Allowing Tester Collaboration 9

ONGOING FEEDBACK 10

- **Ongoing Feedback Objectives** 10
- **Issues** 11
- **Ideas** 13
- **Praise** 14
- **Open Discussions** 15
- **Managing Ongoing Feedback** 16
  - Filtering Feedback 16
  - Filtering Feedback Process 17
  - Scoring Feedback 19
  - Disseminating Feedback 20

DIRECTED FEEDBACK 21

- **Directed Feedback Objectives** 21
- **Surveys** 23
  - Common Surveys 23
  - Product Review Surveys 24
  - Survey Best Practices 25
- **Tasks** 26
  - Task Best Practices 27
- **Additional Types of Directed Feedback** 28
- **Managing Directed Feedback** 29
  - Tester Compliance 29
  - Segmentations in Reporting 30
  - Disseminating Your Data 30

THE LONG TERM VALUE OF GOOD FEEDBACK PROCESSES 31

CONCLUSION 31

INTRODUCTION

The core purpose of beta testing is to collect feedback that can be used to validate and improve a product. That feedback is only useful, however, if it's clear, complete, and properly managed. Otherwise you risk reaching the end of your beta period with a mountain of ambiguous data and no clear plan for how to best use it.

This whitepaper outlines everything you need to know in order to collect and manage impactful feedback during a beta test. By implementing these best practices you will see an increase in both feedback quality and tester participation, and will walk away from your beta test with a comprehensive understanding of what specific improvements will have the greatest impact on your final product.

Is This Resource For You?

> *This whitepaper is primarily intended for individuals running private (also known as closed) beta tests for technology products of nearly any kind, including hardware, desktop software, video games, mobile apps, and websites. This typically includes beta managers, product managers, quality managers, and others tasked with executing a customer beta test in preparation for their product launch.*

"Feedback refers to any information about the product experience collected from your beta testers during a beta test period. This typically includes issues, ideas, praise, open discussions, and other tester-generated data."

The Value of Quality Feedback

Not all feedback is inherently equal. If feedback is confusing, irrelevant, or coming from the wrong people, it could do more harm than good. That's why it's important to not just collect feedback during beta, but to ensure that the feedback is high-quality and actionable. Let's start by defining what we mean by feedback and specifically, high-quality feedback.

Feedback refers to any information about the product experience collected from your beta testers during a beta test period. This typically includes issues, ideas, praise, open discussions, and other tester-generated data.

High-quality feedback is the feedback you can actually use to improve your product. High-quality feedback meets three criteria:

1. **It comes from the right people.** This means the feedback is from objective members of your target market that are not **family, friends, or employees.**
2. **It is relevant to your goals.** Relevant feedback can be used to improve the quality of the product or aligns with the specific goals of your test.
3. **It is complete.** The feedback needs to be clear and have all the context you need to understand the feedback and act on it to make your product better.

High-quality feedback fits all of these criteria, giving you a true picture of the scope, severity, and priority of the issue or idea. For example, a tester could submit an issue saying "The sign-up process didn't work." This is feedback, but not high-quality feedback. For the feedback to be actionable for your team, you'd need additional information, such as what exactly the tester saw that made them think the sign-up process didn't work, the steps that preceded that moment, and the technical details of their environment (i.e., device, browser, OS). These details provide the context needed to accurately assess the issue and take action on it.



High-quality feedback needs to fit three criteria.

Since high-quality feedback is detailed and coming from the right people, it gives you a clear view of how your target market perceives your product. That kind of data will give you the direction and confidence to make meaningful, impactful changes to your product.

The Beta Testing Toolbox

There's a wide variety of ways you can collect feedback from your testers. Some methods (like bug reports or surveys) you may be familiar with, while others (like usage metrics) you might not be. The key is to find and present the right tools to your testers to collect the feedback that meets each of your specific objectives. With the right tools and messaging in place, it's much easier to collect data that you can easily interpret and leverage.

Ongoing and Directed Feedback

In the context of beta testing, we classify feedback into two categories: **ongoing feedback** and **directed feedback**. Each serves a distinct purpose.

Ongoing feedback occurs naturally throughout your test. It's comprised of the continuous insights, responses, and information that your testers report as they use your product. Typical examples are issues, ideas, praise, and open discussions.

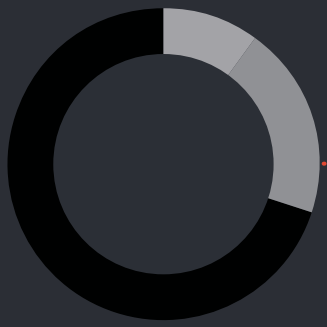
Directed feedback is the result of activities that you specifically request your testers complete at different points during your test. Typical examples include surveys, task lists, or one-on-one calls.

Both ongoing and directed feedback play a fundamental role in the success of your beta test. When used strategically, these forms of feedback can be combined to provide a clear picture of the state of your product, along with meaningful ways to improve it. It's important to remember that different types of feedback collect different kinds of information, and therefore, are necessary to achieve different objectives.

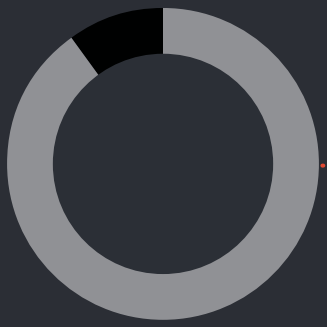
By using a combination of ongoing and directed feedback techniques, a beta manager can collect, organize, and analyze the variety of feedback needed to make meaningful product improvements before launch.

A Note About the Examples Used in this Resource

> *The Centercode Platform is designed to offer a complete beta toolbox. Depending on what tools you're using to run your test, you may or may not be able to leverage all of the advice in this whitepaper. We've done our best to make these best practices as widely applicable as possible, but we will be referencing the functionality of our platform to illustrate many of the concepts discussed here.*



20-30%
AVERAGE BETA PARTICIPATION RATE



>90%
CENTERCODE PARTICIPATION RATE

Feedback Collection Psychology

Beta testers need direction and encouragement throughout a beta test in order to provide the high-quality feedback you need. In a typical closed beta test, the average participation rate is **20 to 30 percent**, meaning that only a handful of your testers achieve the goals you set out for them.

This low level of participation means you'd need to recruit three to five times the number of testers in order to achieve your desired results. You can significantly increase this level of participation (and thus the amount of feedback you collect) by employing **best practices to encourage continued participation from testers.** A skilled beta manager is capable of identifying ideal testers, creating the right environment for high participation, and streamlining the feedback process to gather targeted high-quality feedback.

Many of these best practices come from an understanding of the psychology behind beta management, and specifically, feedback collection. Centercode beta managers typically achieve participation rates above **90 percent** on their beta tests, more than three times the industry average. Through years of experience managing hundreds of tests and many thousands of testers, we've learned numerous valuable psychological principles that should underlie any beta management decisions you make.

Start with the Right Beta Testers

Any good beta test starts with quality beta testers that are joining your test with the right motivations and expectations. For beta tests, your testers should meet three basic criteria:

1. *members of your target market*
2. *enthusiastic about participating*
3. *strangers (not employees, friends, or family)*

In this piece, we assume that you've taken the steps to ensure that you've identified the right testers. Our **Beta Tester Recruitment Kit** *will help you find and identify great testers so you can hit the ground running with an enthusiastic tester team.*

Maximizing Energy Pools and Reducing Friction

Each individual has a different and reasonably fixed amount of energy that they're willing to invest in testing and providing feedback on your product. For some candidates, it will be a lot of time and effort, while others may only be willing to spend a few minutes on your test before moving on to something else. These factors are driven by a blend of their lifestyle (i.e., available free time), personal and professional motivations, and their enthusiasm for your specific product and/or brand.

We consider these varying degrees of commitment as energy pools. As a beta manager, your objective is to gauge and select those candidates with the largest energy pool, and then maximize the impact (i.e., quantity and quality of feedback) of their available energy.

To assess the energy pools of potential beta testers, you need to start with the right recruitment methods. This means building a qualification process that gauges how much time and energy testers are willing to devote to the beta test, so you can select testers with large energy pools. For more details on exactly how to do so, download our **Beta Tester Recruitment Kit.**

After you've selected testers with a lot of energy to devote to the test, your goal is to funnel that energy into providing feedback on your product. The key to maximizing tester energy is eliminating friction in your beta test. Everything a tester does expends energy, with the largest expenditure often being using your product (since the nature of being in beta often produces a frustrating product experience). If you compound this with feedback submission processes that are complex and difficult, your testers will expend valuable limited energy navigating or fighting the system. Based on this principle, it's critical that providing feedback is as frictionless and straightforward as possible.

There are a few simple tricks to reducing friction and maximizing energy with your beta testers.

- **Provide a single centralized system.**
  Your testers shouldn't need multiple user accounts or logins for your beta test. If you have a customer-facing SSO (single-sign on) platform, it's best to leverage that across all beta related resources (e.g., NDA, feedback submission, test information, build access).
- **Clearly set feedback expectations.**
  Educate testers on your feedback systems, so they know how to submit quality feedback. While this process consumes tester energy, the investment will yield substantial results.
- **Never ask for the same information twice.**
  This includes details about their test environment (e.g., mobile phone, operating system, browser) and personal information (e.g., demographics).
- **Never ask for unnecessary information.**
  When possible, you should leverage conditional fields to lead testers through relevant data collection.

Following these specific best practices greatly increases both the level and quality of your tester feedback.

Ultimately, it's very easy to use up significant amounts of a tester's energy pool on trivial requirements or inconvenient processes. If testers are trying to submit an issue looking up their router model number (again), or need to log into different systems to submit their feedback, that's energy that isn't going toward using your product or providing valuable feedback. It's your job as a beta manager to ensure this isn't the case.

Validating Your Beta Testers

In every beta test there's a natural **feedback loop.** It's a simple but powerful process:



The feedback loop ensures that the conversation between you and your testers isn't a one-way street.

The vast majority of testers **aren't motivated by free products and incentives** but are instead drawn to beta testing for the opportunity to contribute to and improve a product they use. This means that your testers are naturally excited about helping you improve your product. What can turn them off, however, is if they feel their contribution isn't recognized or appreciated by you or your team.

Many beta managers simply collect feedback without responding to testers and closing the feedback loop. This can leaves testers feeling like their feedback is going into a black hole, which will result in decreased participation rates and lower quality feedback. Thus, closing the feedback loop by letting testers know that their feedback was received and is appreciated (ideally within one business day) plays an important role in maintaining continued tester participation.

Feedback responses don't need to be complicated. They can be as simple as a quick line letting testers know you've read their submission and thanking them for it. If you have the information, you can even tell testers what's being done to fix the issue and let them know they might be asked to test the fix later in the test. You can also help the tester by giving them a workaround to their issue in the meantime. These small responses provide crucial validation for your testers and make them feel like they're a part of the product improvement process. It lets them know they're making a difference and that you're listening to what they have to say. By doing so, you encourage testers to give better, more robust feedback as your test progresses.

Don't Automate Tester Validation

It's tempting to automate your thank you messages for tester feedback (especially if your beta test is getting a lot of submissions), but this can backfire. If testers see the same template response to every piece of feedback they will quickly get a sense that the response isn't genuine. This can negatively affect their participation because they no longer feel validated and appreciated. Take the time to write unique and real responses to your testers. They will pay you back tenfold with increased energy and feedback.

Setting Participation Expectations

A common mistake new beta managers make is assuming testers instinctually understand what they're supposed to do during a beta test. In truth, most testers (even the naturals) require guidance. It's important that with everything you expect testers to do, you provide the necessary direction and support to do it.

It's critical to clearly share your expectations with your testers. This means making certain that they understand what they're supposed to do, and how often you would like them to do it. You should set these expections early in the beta test, such as in a welcome email or intro call. You should also provide written resources testers can reference throughout your beta test about how to use your beta management tool and generally how to be a successful and valuable tester.

As part of this you need to make sure that your participation expectations are reasonable and align with what testers can deliver. For example, you want testers to submit issues as they discover them. Some testers will discover a plethora of issues, and some won't find any. So setting a participation expectation that each tester will submit five issues during your test is setting unreasonable expectations and asking testers to invent problems.

Instead, you should tell your testers that they're expected to actively use the product as intended and log all issues and ideas as they go. Then you can focus your participation requirements on activities that are more easily measured, such as completing all assigned activities within five days. These are requirements that all testers should be able to meet, even if they don't come across any issues.

Collecting a Variety of Feedback

Your testers will have a wide variety of feedback to provide about your product. They will want to tell you about problems they encounter, ideas for improving the product, and details about how it fits into their lives. If you only have one way for testers to provide feedback (e.g., a bug report form) then one of two things will happen. Either testers will submit all of their feedback through that single outlet (cluttering your data) or they won't submit many of their thoughts, meaning you'll miss out on incredibly valuable insights that would otherwise be free.

By giving your testers numerous ways to engage (e.g., issues, ideas, praise, open discussions), you're both increasing the breadth of your data while making it easier for you to process and leverage it.

Multiple Feedback Types Increase Participation

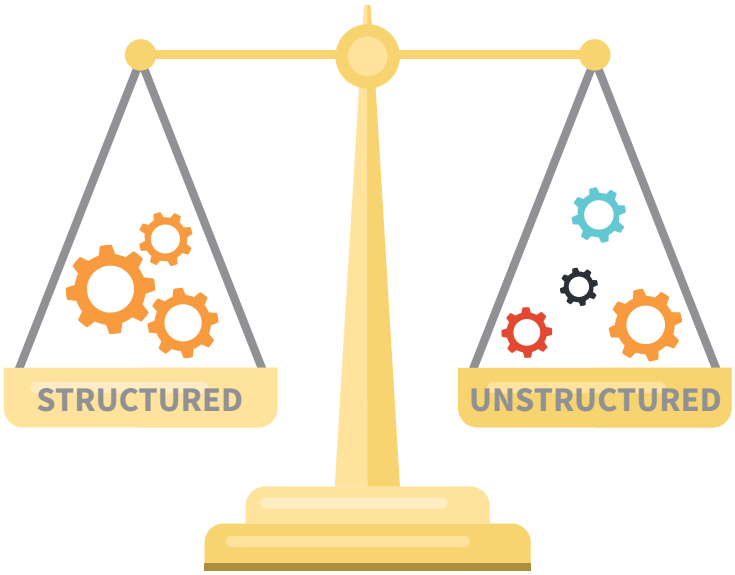
Some companies don't collect feedback like ideas during beta testing because they don't have immediate plans to leverage that data. Their thought is that they should focus testers on only the types of feedback that are most valuable at the moment. Aside from keeping your data clean, collecting these types of feedback serves a psychological purpose. It makes your testers feel like they're being heard and valued — as opposed to just being crowdsourced quality testers. By allowing testers to submit all of their feedback, you will increase participation and feedback in other areas of your test that you do care about (such as issues). So even if you don't have immediate plans to leverage the data, it can still serve a positive psychological purpose to collect it.

Balancing Testers' Activity

In every beta test you need to strike a balance between allowing testers to use the product as they naturally would in the real world and giving testers assigned activities to complete. The specific balance you aim for should be relative to the unique objectives of your test.

Unstructured usage provides important information about how testers naturally interact with the product. This can be critically important to understanding user acceptance and exposing outlying issues that would likely be missed in traditional focused quality testing.

Structured activities ensure coverage of all elements of the product and give testers a good starting point for their feedback.



You need to strike a balance between structured and unstructured activity. This will help you achieve a variety of goals while increasing the amount of feedback you receive.

It is often useful to start with a basic set of structured activities (such as an out-of-the-box survey) intended to kickstart tester engagement. Beyond this, testers should be encouraged to explore further for a reasonable amount of time. Additional structured activities should be spread throughout the test to ensure each unique objective or feature area is covered.



If you only have unstructured activity, then you're relying on testers to find their way around your product, which may not give you the full picture of the state of your product.



If you overload your testers with activities, then they could become frustrated that they aren't getting to use the product like they want to, decreasing participation.

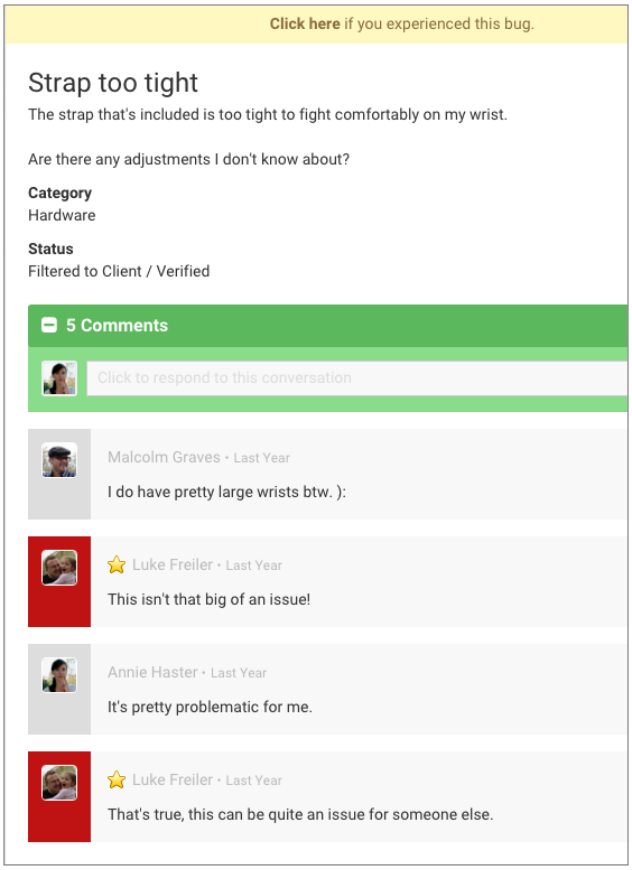
Allowing Tester Collaboration

Collaboration plays an important role in collecting high-quality feedback during beta testing. Traditionally, most feedback in a beta test has been a private, two-way conversation between a beta tester and a beta manager. The beta tester submits an issue, the beta manager asks for any additional information (if needed), and then the beta manager processes the issue. The problem is that this only gives the beta manager a single beta tester's perspective, which lacks important information about the scope and frequency of the issue.

We recommend allowing testers to see and collaborate on each other's feedback during a beta test. Giving testers the chance to discuss and vote on feedback does three important things. First, it gives you a clearer, cleaner picture of the issue being discussed because all of your testers are contributing their experiences to a single conversation. You can see which testers are running into the same issue and which ideas are the most popular, giving you a more complete picture.

Second, it increases confidentiality by giving your testers a controlled space to talk with other testers about their excitement and user experiences. Funneling testers' excitement into private channels where they can safely chat with other beta testers makes it less likely that their excitement will leak onto public forums or social media. It also allows you to capture their conversations in your beta platform, where you can analyze them for trends.

Third, letting beta testers talk with each other increases their participation and engagement. They feel like they're part of a team, working towards a common goal. You'll find that testers will jump in to help a comrade find a novel workaround to an issue, or try to reproduce an issue someone else submitted on their own beta unit. This sense of camaraderie will give you a stronger, happier beta tester team, resulting in higher quality feedback.



Allowing testers to view and contribute to each other's feedback provides a more complete picture of the issue.

Collaboration Might Not Be Right For You

While we recommend allowing collaboration and discussion between your beta testers, it might not make sense for your beta test. That decision depends on your policies, audience, product, objectives, bandwidth, and system capabilities. If your situation isn't conducive to allowing collaboration between your beta testers, you can still use most of the feedback collection methods discussed in this whitepaper; you'll just skip the parts that involve collaboration. You'll also want to focus additional attention on communicating individually with your testers to keep them participating.

ONGOING FEEDBACK

A large part of the feedback you'll collect during your test will be ongoing feedback. As each tester experiences your product, they will have issues or ideas about your product that will naturally arise. Testers will run into issues, like or dislike certain features, or want to discuss aspects of the product that could be improved. Given the organic nature of this feedback, you'll need pre-determined processes in place to collect, triage, analyze, and prioritize this feedback. That way, as your testers expose more about their feelings and experiences with your product, you'll begin to amass a healthy amount of usable, high-quality feedback to inform your imminent product decisions.

Ongoing Feedback Objectives

There are four **basic types of ongoing feedback,** each of which inherently achieves a few common beta testing objectives.



ISSUES

Test quality, compatibility, and real-world performance



IDEAS

Shape product roadmap and measure customer acceptance



PRAISE

Gather delights, product strengths, and real-world insights



DISCUSSIONS

Emulate real-world discussions and drive focused collaboration

Since each feedback type achieves unique objectives, we include all four of these feedback types in every beta test we run. This ensures we collect feedback that both provides testers numerous channels to submit varied feedback and achieves a diverse set of useful objectives.

Once you understand the objectives that each feedback type achieves, you can design forms and processes to make the most of each one. Over the next few pages, we'll dive into how to make the most of each of these types of ongoing feedback in your beta test.

Keep in Mind How You Will Use Your Data

As you're building your forms, keep in mind how you're going to report on and process your data. Rating scales are much easier to report on than text boxes. Dropdown options make it much easier to trigger workflows than open-ended descriptions. Understanding how you're going to use each field in your forms will ensure that you aren't asking for unnecessary information and that you're asking for information in a format you can use.

Issues (aka Bugs, Defects)

A beta test gauges how your product will perform in the real world. This is most likely the first time your product will be in the hands of real customers. Your product will be tested in more ways and environments than you could ever realistically emulate in a lab. As a result, a plethora of both known and unknown issues will be revealed throughout your beta test. Creating a comprehensive but easy-to-use form for reporting issues will help you collect the information your quality team needs to assess, duplicate, and fix these issues.

When building your issue report form, you need to balance simplicity with completeness. You want to make it easy for a tester to submit an issue, but make sure you get enough information so your team can reproduce and fix it.

At a minimum, feedback forms for issues should include:

1. **Test platform:** This field allows the tester to attach a detailed device profile to their issue submissions. Before the test begins, testers fill out detailed information about the devices they own. For example, they would provide the details of their smartphone before a mobile app test. Then that context is attached to their feedback without having to provide it each time. If you're using a different beta management platform, you'll need to include form fields that capture this context.
2. **Summary:** This will act as a title and allow you and other testers to understand the issue at a glance.
3. **Feature:** These categories will be defined by you before the test and will align with the different elements/features of your product. The tester then assigns a feature when they submit an issue, so you'll know what part of the product the issue affects.
4. **Steps to reproduce:** This field allows the tester to explain exactly what they did leading up to the issue and what happened when it occured. This will make it easier for your team and other testers to reproduce the problem. Seed the text box with step numbers (1, 2, 3, 4) so your testers know to provide specific steps. Then include the text "Tell us what happened:" to make sure they also explain what they encountered when the issue occured.
5. **File attachments:** This is a place for the tester to attach any screenshots, crash logs, videos, or other files that could help your team understand the issue.
6. **Blocking issue:** We ask the tester "Is this issue stopping you from further testing?" This means that the issue they've encountered has compromised basic functionality or has completely stopped them from using the product and providing feedback. This will flag the issue so the beta team can provide immediate support to the tester.

Known Issues

You probably have a list of known issues going into your beta test that your testers could run into. You have a few options when handling these. First, you could not mention them and see how many testers run into them. Second, you could provide your testers with a list of known issues so they're informed. Third, you can seed your issue forms with the known issues so testers can submit those issues just as they would if another tester submitted the it. How you approach it really depends on the known issues and how helpful additional context from your testers would be in resolving them.

Issues (aka Bugs, Defects)

ADDITIONAL FIELDS

Beyond these fields, you may need or want to include additional fields, depending on your situation. For example, we don't ask testers to provide a severity rating for an issue because we find ratings by our beta team to be more reliable (which we'll discuss momentarily). You can ask testers to assess how severe they feel the issue is, so you can prioritize accordingly, but we suggest you pair this field with an internal severity rating.

If you do choose to add additional fields to your issue report form, make sure you're only asking for information that's important to achieving your goal of understanding, reproducing, and prioritizing feedback while supporting the tester. Every unnecessary field introduces friction that limits participation and decreases feedback.

INTERNAL FIELDS

You'll need to include a few hidden fields on your feedback forms that will allow you to process and manage tester submissions, but that don't need to be visible to testers. The three internal fields the Centercode team uses are Severity, Reproduction, and Status. After a tester has submitted an issue, a member of our team will assign the issue's Severity based on the information the tester has submitted (we've found that testers typically lack the context necessary to provide objective ratings on their own). We will then attempt to reproduce the issue in our lab and indicate whether we were successful. Finally, we use the Status field to indicate to our team where the issue is in our workflow by using the following statuses: new, needs more information, closed, or sent to team. We also have a system for marking duplicate feedback in our system, but you could use a status to do so as well.

COLLABORATION

We allow our testers to see and contribute to each other's feedback throughout the test. For issues, testers can contribute in a couple of ways. First, we allow them to review submitted issues before completing a new feedback form, so they can indicate if they're running into an issue that's already on the beta team's radar. Second, they can comment on an existing feedback form to provide additional context that they feel is missing. Third, they can opt to try and reproduce an issue that's already been submitted to help the beta team see how widespread an issue is. All of these forms of collaboration give the beta team important context to bugs and defects.

Since we encourage collaboration on these reports, we also include a message at the top of all of our forms that reminds testers that their feedback will be visible to other testers. We ask them to write clearly and use good grammar so that other testers can easily understand what they're communicating. This gentle reminder has made a notable difference in the clarity of our submitted feedback.

Crash Logs and Other Automatically Generated Data

Chances are your product is generating some back-end data as your beta testers use it. This could include crash logs and other quality-related data that can help you improve your product. Consider how you want to connect this data to your submitted issues and then educate your testers accordingly. We've seen companies have testers attach screenshots of their crash logs to the issue or copy and paste the logs into their form. They've also provided directions for testers to submit logs straight from the device. However it works for your product, make sure the testers understand what's expected of them so you can use this data to provide additional context.

Ideas (aka Feature Requests, Suggestions, Requests for Enhancement (RFEs))

Ideas allow you to collect information about what testers would like to see in your product. This can help influence your roadmap and gauge user acceptance of the current design. As with issues, you need to balance ease of use with completeness when creating your idea submission forms. They should include the following fields to get the full image of what the tester is imagining for your product:

1. **Summary:** A short summary will allow you and other testers to understand the idea at a glance.
2. **Feature:** These categories should be the same as the ones for issues. This allows the tester to indicate the part of the product their idea involves. This will allow you to process ideas and suggestions more efficiently.
3. **Description:** This large open text box will allow the tester to provide a detailed explanation of what ideas they'd like to see in your product.
4. **File attachments:** This optional field allows your testers to submit any files (screenshots, mockups, etc.) that could help illustrate their suggestions.

ADDITIONAL FIELDS

If you feel there are other pieces of information you need to understand the idea, you can include those in the form. The most popular additional field we've seen is to allow testers to rate how important the feature is to their user experience. As we mentioned before, just make sure you keep the required fields to a minimum so the submission process isn't discouragingly long. A good idea submission form allows a tester to submit a vague idea, as well as very specific improvements for your product.

INTERNAL FIELDS

Just like with issues, your idea submission forms should have internal fields that your team can use to manage your feedback, but testers cannot see. With ideas, the only internal field we use is Status, and we have the same available statuses you saw with issues: new, needs more information, closed, and sent to team. We also have the same duplicate management tools that allow us to manage duplicate submissions on the back end.

COLLABORATION

As with issues, we allow our testers to collaborate on all ideas. This means that testers can vote on other testers' submissions and use the comment logs below each idea to help flesh it out or simply contribute to the conversation. This helps popular ideas rise to the top, making it easier to prioritize ideas later.

Optional and Required Fields

Remember that not all of your fields will be required. Review your forms after you build them to make sure you're only requiring the fields you truly need. With the idea submission form we outlined, for example, all fields are required except the file attachments field. That field is only necessary for testers that would like to provide file attachments for additional information. If you made all fields required, testers would be forced to provide file attachments even when they don't feel they're necessary, introducing friction and frustration.

Praise (aka Positive Feedback, Delights, Kudos)

Not all feedback is negative. Specific feedback about positive experiences with your product from testers is very valuable. You can use tester praise to give kudos to the development and UX teams. Beyond just making those teams feel good, they'll have a better idea of what is working well and how to make more of the same. You can also use praise in your marketing messaging as customer testimonials and in sales conversations to highlight delightful product areas.

The form you use to collect praise from testers should, like other feedback types, be easy to submit while providing enough context to take action if you so choose. Your praise form should include the following fields:

1. **Summary:** This field is a large text box where testers can share how they used the product that day and what they liked about the experience.
2. **Feature:** These categories should be the same as all other feedback forms. This allows the tester to indicate the part of the product the praise involves, which allows you to process the submitted feedback more efficiently.
3. **Description:** This large open text box will allow the tester to provide a detailed explanation of what praise they'd like to give about a specific experience with your product.
4. **File attachments:** This optional field allows the tester to include any files (screenshots, video, etc.) that could help illustrate their praise.

ADDITIONAL FIELDS

If you feel there are other pieces of information you need to understand the positive feedback, you can include room for those in the form. The most popular additional field we've seen is to allow testers to rate their delight with the user experience. As we've mentioned before, just make sure you keep the required fields to a minimum so that the submission process isn't discouragingly long.

INTERNAL FIELDS

Just like with issues and ideas, your praise forms should have internal fields that your team can use to manage your feedback but that testers cannot see. With praise, the only internal field we use is Status, and we have the same available statuses you saw with issues: new, needs more information, closed, and sent to team. We also have the same duplicate management tools that allow us to manage duplicate submissions on the back end.

COLLABORATION

As with issues and ideas, we allow our testers to collaborate on all praise. This means that testers can vote on other testers' praise and use the comment logs below each praise submission to contribute to the conversation. This helps popular praise rise to the top, making it easier to prioritize later.

Open Discussions (aka Private Forums)

Along with forms to submit issues, ideas, and praise, you want your testers to have a controlled place for general discussions about your beta product. This will allow you to capture customer sentiments that aren't easily categorized as an issue or idea.

Similar to your other types of ongoing feedback, you'll need a form for testers to start a discussion. Your open discussion form should have the following elements:

1. **Topic:** This field allows the tester to quickly say what topic they'd like to discuss.
2. **Feature:** These categories should be the same as all other feedback forms. This allows the tester to indicate the part of the product the discussion involves, and allows you to categorize discussions accordingly on the back end.
3. **Body:** Here, the tester can provide a more detailed description of the subject matter.
4. **File attachments:** This field gives your testers the option to submit any files (screenshots, pictures, etc.) that are relevant to the subject being discussed.

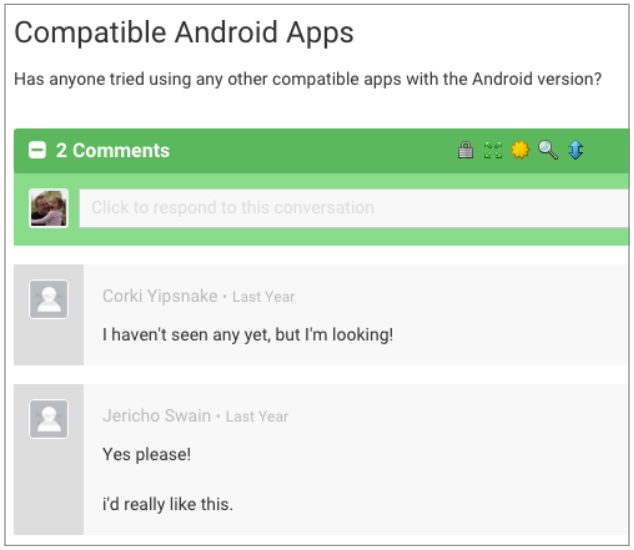
INTERNAL FIELDS

As with ideas, open discussions use an internal Status field that allows the beta team to categorize the discussion based on our workflow. We have the normal statuses (new, needs more information, sent to team, and closed), but also have a "reviewed" status for discussions that don't necessarily require additional action but that your team has reviewed.

COLLABORATION

These discussion boards are a classic way for testers to channel their excitement about the product into productive discussions with other testers. It's also a great chance for beta managers to engage with testers and encourage their participation. Savvy beta managers will be able to pick up on themes in the discussions that could inspire future surveys or tasks to get more structured feedback on relevant topics.

You can also seed your beta tests with specific discussions you'd like to see, such as asking what testers think about the UI color palette. These prompts will give testers a launching-off point for discussions and spark additional participation and product exploration.



Discussions give testers a controlled environment to share their excitement and thoughts about the product.

Custom Feedback

These four types of ongoing feedback may not be the only ones in your beta test. We've seen our customers get incredibly creative with their feedback forms: collecting videos, images, and even exercise logs if that's what they need to improve their product. Before your beta test begins, consider whether there's any ongoing information you need to collect that isn't covered by the forms discussed here.

Managing Ongoing Feedback

Collecting feedback is just part of the puzzle. Effective management of your ongoing feedback is just as important as the raw data you're collecting. Creating processes for handling your feedback goes a long way toward making sure it's used to improve your product. It takes careful management, both during and after a beta test, to maximize your results.

There are two parts to managing your ongoing feedback:

1. Part one consists of cleaning, triaging, and prioritizing feedback in real time during your test. A good beta team will constantly work with testers to get clear and complete feedback from them while prioritizing that feedback based upon pre-planned criteria.
2. The second part of ongoing feedback management has to do with what you do with the data after it's been cleaned and scored. As you disseminate all the feedback you've collected, it's important that you send it (either automatically or manually) to the right systems and members of your team, with the right context.

Filtering Feedback

As testers submit their ongoing feedback during a test, your team is going to read and react to that feedback. Your goal is to make sure the feedback is as clear and complete as possible before sending it to the correct person at your company (e.g., QA, product management, marketing). To do so, you want to review the feedback for a few important qualities.

In our beta management system, we have status and workflow functionality that makes organizing ongoing feedback easy. You can use statuses to process ongoing feedback and duplicate management features to organize similar feedback without losing information. If you don't have these features available in your system the filtering steps on the next page will still apply, but you'll have to adjust your responses accordingly.

Feedback Filtering Process

This is the **feedback filtering process** we follow for every piece of ongoing feedback we receive during our beta tests. At the end of this process, you will have high-quality feedback to send to your team.

1

Validate Feedback

Is this the correct type of feedback?

If the feedback type is incorrect (e.g., an issue, should be an idea, beta portal problem, general venting), direct tester to appropriate place and close the issue.

2

Confirm Originality

Is this a known issue (previously reported or internally recognized)?

If previously known, bind feedback to the original issue.

3

Confirm Clarity

Is the message the beta tester is attempting to communicate clear?

If the message is unclear, request additional information from the tester. If the tester doesn't respond, remind them a few times before closing the issue.

4

Polish Text

Is the feedback well written and easy to read?

Fix obvious spelling, grammar, capitalization, and punctuation issues to increase readability of the feedback.

5

Verify Feature

Is the tester's Feature selection accurate?

If incorrect, select the appropriate Feature.

PROCESS CONTINUES ON NEXT PAGE

6

IS THIS AN ISSUE?

YES

6a

Set Severity

How impactful is the issue?

If you have an internal field on your forms for Severity, select the appropriate Severity based on your Severity guidelines.

6b

Reproduce

Can the issue be reproduced by the beta management team?

Attempt to reproduce the issue. If reproducible, note it on the report. You can also add a comment encouraging other testers to attempt to reproduce it and monitor their responses.

NO

Blocking Issues

Blocking issues are a special circumstance in which an issue prevents a participant from further testing. While rare, it is critical that these issues are managed as quickly as possible because until the issue is resolved, that tester cannot contribute to your beta test. Identify a technical lead at your company who will be available to help testers with major technical problems they encounter during a test. If a tester submits a blocking issue, attempt to validate the issue, then loop in your technical lead to help you support the tester and find a solution so they can continue testing.

7

Thank and Encourage

Would peer contribution add value?

Add a comment to recognize the issue and provide positive feedback to the tester.

Make Public

Are we ready for open collaboration?

Change the feedback to public so that other testers can see it. In our beta tests, we only make issues public after review. Ideas and discussions are public by default.

8

Send to Team

Is this feedback original, clear, and ready to move on to the appropriate stakeholders?

Notify the appropriate member of your team (QA, support, product management, marketing) that there's relevant feedback for their review.

9

Scoring Feedback

As your feedback rolls in you will need a way to prioritize tester submissions. Otherwise, all ongoing feedback will jumble together, making it difficult to do anything with it. The best way to keep track of what's coming in is to create a scoring system that will allow you to assign certain degrees of importance to different aspects of your feedback. You can then combine this with the popularity of that feedback to help you prioritize and handle it accordingly.

By assigning weights to different aspects of your feedback, the most important feedback will rise to the top. Use a weight of 1.0 as the baseline and then adjust up or down based on the importance of the attribute. For example, an issue is more important than an idea so an issue would have a weight of 1.5 and an idea would have a weight of 0.8. Furthermore, a critical issue is more valuable than a cosmetic one, so give a critical issue a weight of 2.5 and a cosmetic one 0.5. By combining these weights, the more important feedback becomes easy to pick out.

We assign different weights to each element of the following aspects of our feedback:

- **Feedback Type**
- **Feature**
- **Severity (issues only)**

In addition to looking at the innate aspects of a piece of feedback, you should also take into consideration the popularity of a piece of feedback when calculating its score. Our system combines the following factors when calculating the popularity score of a piece of feedback:

- **Duplicates** - How many times was the same issue submitted by different testers?
- **Votes** - How many testers indicated that they had the same issue or opinion as the submitter?
- **Comments** - How many of the testers contributed to the discussion?
- **Viewers** - How many testers looked at the feedback?

Our system uses an algorithm that combines the feedback score and popularity score for each piece of feedback and then organizes it, with the highest rated pieces on top. These are the pieces of feedback that will have the most impact on your product. This will help you make sense of the pool of information coming from your beta test and determine where to focus your team's limited resources to have the largest impact on your product before launch.

Rank	ID	Summary	Severity	Feature	Comments	Occurrences	Score
1	TP-0102	Security camera reboots when accessed remotely	Major	Remote Access	41	24	190
2	TP-0050	Export video clip button shows as disabled but still works	Major	User Interface (UX/UI)	19	17	152
3	TP-0150	30 second video clip is only displaying 26 seconds of video	Major	Video Clip	17	13	116

Automated scoring allows your most important feedback to rise to the top.

Disseminating Feedback

Once you have clean, prioritized data coming in, you need to make sure that feedback gets in front of the right people on your team so they can use it to improve your product.

WHO WILL BE INSIDE THE BETA?

All of your feedback will be coming in through your beta management system, but not all of your company will have access to that system. Decide who from your company will be part of your beta test and accessing feedback directly. At the very least, it's helpful to have a technical lead (likely from your QA team) who can see the issues coming in and support testers facing blocking issues. However, if there are other teams (such as product management, support, or marketing) that are heavily invested in the beta, they may want to have a representative in the beta as well to work with testers and make sure their goals are met.

WHAT NEEDS TO GO WHERE, WHEN?

Much of your data will need to be disseminated outside of your beta management system. This means building predictable workflows to send that data to the right people, in the right way, at the right time. To do so, you need to determine what data needs to go where (into which systems), when. For example, your head of QA may want all critical issues sent into JIRA immediately, but just a report of the most popular issues emailed to them once a day. Your product manager might be okay with waiting until the end of your beta test to receive a prioritized list of all of the ideas.

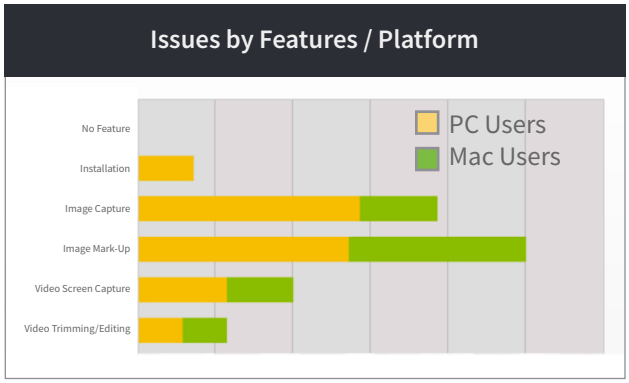
You also need to make sure your feedback gets to your team with the right context. If your QA team only sees the description of an issue and the steps to replicate it from the initial report, they're missing a lot of valuable context. Make sure you're either sending them the pertinent information (such as test platform, feedback score, and tester discussion) or giving them access to that information in your beta management system.

No matter what reports you decide to send, put the processes in place before your beta test begins. While you can create reports and send them to your colleagues during your beta test, you'll have a lot of things vying for your attention at that point. Most tools allow for automatic report creation and dissemination, which can save you a lot of time once your beta is underway.

If you're not careful, the demands of ongoing feedback can overwhelm you and lead to important issues falling through the cracks. Thinking about who needs to see what data (and when) will help you make sure all the relevant information gets on your team's radar at the right moment.

Weekly Reports

Each of our tests includes a weekly report that gives relevant stakeholders a quick overview of what's happening in the beta test. We include key metrics in the test for that week, including the top pieces of ongoing feedback, and charts showing the breakdown of feedback by feature, severity, and other relevant segmentations. This can be set up before your test begins to keep all the relevant stakeholders in the loop once the test is underway.



Weekly reports can highlight the most important discoveries in an ongoing beta test.

DIRECTED FEEDBACK

The second type of feedback in a beta test is directed feedback. These are the activities and questions you directly ask your testers to do or answer during your beta test. The two most commonly used kinds of directed feedback are surveys and tasks, but this feedback can take many different forms. Directed feedback plays a crucial role in beta testing, because it allows you to get specific data from your testers to meet your objectives, rather than just hoping that information comes up as testers use your product.

Directed Feedback Objectives

A beta test can accomplish virtually any objective. That's why your beta test has to be built around fulfilling your specific goals. While ongoing feedback inherently achieves certain objectives (such as testing product quality and gauging user acceptance), directed feedback can achieve any objective. If you want to assess the installation process, you can write a survey to do so. If you want to test firmware updates, you can assign your testers a task to update their firmware. Directed feedback gives you the flexibility to achieve a wide variety of goals. The question you then need to answer is: what goals would you like to achieve, and what form(s) of directed feedback will get you the appropriate data to achieve those goals?

To determine the directed objectives you'd like your beta test to meet, ask yourself a few questions:

1. **What would you like your testers to do?**
2. **What questions would you like this beta test to answer?**

Answering these questions will give you an idea of what activities you need to design for your testers. If there is a specific feature that's new or particularly troublesome, set a directed objective to have testers focus on that feature.

If you're having trouble determining your objectives, another way to think about it is: What's keeping you up at night? If you can answer that, then you'll know what your beta test needs to accomplish. Here are a few of the most common objectives we see directed feedback achieving in our managed beta tests:

- **Test** the installation or out-of-the-box experience.
- **Assess** the quality and/or user experience of specific product features.
- **Regress** fixes for solved issues.
- **Compare** preferences for different options or features.
- **Assess** customer/feature acceptance over the course of the beta test.

You don't want to have too many directed objectives, otherwise you'll overload your testers with surveys and tasks to complete. We recommend having no more than one directed objective per week. This will allow you to maintain balance in your test. When you're brainstorming your directed objectives, rank them in order of importance. This will make it easier to decide which ones to include if you don't have time to cover them all.

When planning your directed objectives, also keep in mind that you may need to use multiple activities to reach a single objective. For example, you might assign testers a task to update their app to the latest version, then give them a survey about their update experience. You could also achieve multiple objectives (or parts of multiple objectives) with a single activity. For example, you could have testers complete a survey about their initial impressions of the product, which could assess the out-of-box experience and user acceptance of certain features.

Once you've determined your objectives, the next step is to decide which types of directed feedback will help you achieve those objectives. There's a variety of ways you can collect directed feedback, each of which has specific qualities that make it unique and valuable. You need to consider these qualities when deciding which activities make the most sense for your beta and its specific goals.

There are two popular types of directed feedback that you should incorporate into your beta test: surveys and tasks.



Surveys

A survey is a list of questions you give your testers to measure user insights, beliefs, and motivations regarding their experience with your product. Surveys are valuable when you're looking for quantifiable data about your testers' opinions about your product and the user experience.



Tasks

Tasks are assigned activities you ask your testers to complete during your beta test. Tasks are useful when you want to focus testers on a specific piece of your product. This can be a new feature or a particular aspect of the user experience that you plan to survey them about later (such as the onboarding experience).

Using Directed Feedback to Increase Participation

As a side benefit, directed feedback also helps keep your testers engaged. Assigning testers tasks to complete will encourage product usage that could result in more issue or ideas submissions. Asking testers to complete a survey might encourage discussions amongst testers on your forums. Just make sure you don't overload your testers with activities or they won't have time to explore the product on their own.

Surveys

Surveys are probably one of the first things people think of when they think of beta testing, and for good reason. They're one of the most commonly used forms of feedback in beta testing. Surveys are used in just about every beta test because they're a straightforward way to collect quantifiable data that can point to trends amongst the beta users.

Surveys provide quantifiable data about the user experience from your testers. You can gather tester sentiments about everything from the installation experience to the ease-of-use of specific features. You can use this data to look at the general reaction users had to your product, or slice and dice the data based on specific segmentations, such as age or platform. Because all of your testers answer the same questions with a survey, they provide a powerful preview of how your overall target market will react to your product once it's available in the market.

As effective as surveys can be, it's important that you don't overuse them. If used sparingly they can boost participation and product usage. However, if you overload testers with required surveys it will take time and energy away from their natural use of the product, which will affect the amount of ongoing feedback you receive. It could even cause your testers to rush through the surveys, giving you skewed or useless data. Unless absolutely necessary, don't assign more than one survey a week. This will strike the balance in between directed and ongoing feedback.

Common Surveys

You can build a survey around just about anything (a goal, a feature, an issue); it simply depends on what you're trying to accomplish. Here are the surveys we see most often:

- **First Impressions Survey**
  This survey is given to testers at the very beginning of a test and covers any unboxing, onboarding, or installation processes testers went through. It should also ask about their initial impressions of the product.
- **Feature-Specific Surveys**
  These surveys ask testers detailed questions about their usage of and opinions about a specific feature.
- **Feature Usage Survey**
  This survey lists the features of a product and asks testers which ones they've used to assess coverage and popularity of certain features.
- **Weekly Surveys**
  These surveys check in with testers on a weekly basis to assess their experience with the product that week and ask standard questions that track customer acceptance metrics over the course of the test.
- **Task Follow-up Surveys**
  These surveys are given to testers after they've completed a task (or tasks) to get more detailed information about their user experience.
- **Product Review Survey**
  These surveys ask the tester to rate the product overall and then asks for explanations of their ratings. We go into more detail on this survey later in the section.
- **Final Survey**
  This survey will be the last activity your testers complete during your test. It looks at the big picture to see what testers thought about your product features and the user experience.

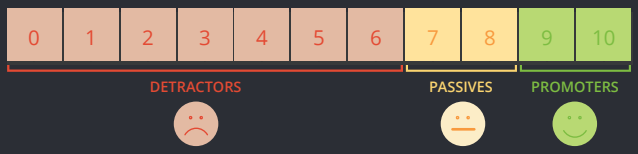
Product Review Surveys

We include one standard survey at the end of every single test we run and it provides a powerful indicator of how the product would perform in the market in its current state. Our product review survey uses two standard rating methods for products to illustrate the strengths and weaknesses of the beta product.

Net Promoter Score (NPS)

The first question in our product review survey asks how likely a tester is to recommend the product to a friend or colleague on a scale of 0 to 10. Take the percent of people that give a 9 or 10 and subtract the percent that gave a rating of 0 to 6 to get the product's **Net Promoter Score (NPS)**. *NPS is a commonly used benchmark to measure customer satisfaction on a scale of -100 to 100. NPS is used widely enough that you can compare the NPS of your product during beta with the NPS of other products at your company or in your industry. Along with the NPS rating we ask testers to explain why they gave the product the rating they did. This provides useful context about the parts of the product that are leaving the best (and worst) impressions on the users.*





Star Rating

The second question we ask simulates a product review like a customer would find on Amazon. We ask testers: "On a scale of 1 - 5 stars, how would you rate this product if you had purchased it from a retailer?" Then, depending on the star rating they give, we ask a follow-up question to pinpoint exactly what about their experience led to that rating. This provides useful information about what improvements could make the most impact on the product.

Using standard survey questions can provide valuable benchmark data throughout your beta program. You can use them to gauge testers' opinions about your product over the course of your beta test to see how perceptions evolve. You can use them as standard metrics to compare different products within your company or different releases of a product to see if it's improving over time. The idea is to use these standard measurements to mimic how the product could do once it's released to the public.

> *"On a scale of 1 - 5 stars, how would you rate this product if you had purchased it from a retailer?"*



Survey Best Practices

There are hundreds of books written about survey writing and analysis. Poorly written surveys will give you useless or misleading data. Overly long or complex surveys will burn out testers and give you poor results. While we can't cover all the ins and outs of survey writing here, we've put together our top advice for good beta surveys.

- ✓ **Keep surveys quick and focused.** In most scenarios, testers are volunteering their time and energy. Respect that. Generally, 10 questions is a good survey, 15 is long but acceptable, and 20 is only really appropriate at the end of a beta test (since you won't be asking for much more afterward). If you plan to survey your testers more than once a week, keep them to around five questions each. Before you start writing your survey, ask yourself "what do I want to know?" Focus on gathering the data you need to answer your question and avoid adding in a bunch of "nice to know" questions that will just make your survey longer and more tedious.
- ✓ **Determine the target audience for your survey.** Not every survey needs to go to every tester. Maybe you only want testers who are tech-savvy to answer your survey. Maybe you only want the opinions of testers who have successfully used a certain feature. Asking all of your testers everything could cloud your data with irrelevant responses.
- ✓ **Remove bias and confusion from your questions.** How you ask a question makes a big difference in how useful your data is. When writing your questions, make sure you aren't including leading language (e.g., "How easy was the product to use?") or asking multiple things in a single question (e.g., "Rate the intuitiveness of the hardware's setup and use.").
- ✓ **Keep questions short and the words simple.** The shorter your questions are, the easier they will be for your testers to understand and answer. It will also be easier for you when you're creating graphs and reports. If your questions are longer than one line, consider rewording or even revisiting if you're trying to cover too much in the question.
- ✓ **Think about how you want to use the data when crafting the question.** What question are you trying to answer? Do you need to be able to compare the responses to each other or to a baseline? Do you want to know which device testers primarily use to watch movies, or if they use any of the devices listed? Small wording changes can make a big difference, so make sure the questions are collecting the data you really need in a way you can use.
- ✓ **Use rating scales of 5 (not 10).** Although common, there is no reason rating scales need to be from 1 to 10. Rating scales with 5 points are much easier for both testers and your team. A 5-point rating scale allows room for strong feelings (1 and 5), general good or bad feelings (2 and 4), as well as indifference (3). This makes selecting choices more natural and obvious, while also making reporting easier and cleaner.
- ✓ **Label your rating scales appropriately.** Rating scales are useful in nearly every survey. Unfortunately, many surveys have unmarked values (1, 2, 3, 4, 5), which can be interpreted differently by every tester. By giving labels to the first and last values (such as 1=Strongly Disagree, 5=Strongly Agree), testers are given a clearer picture of what the values are intended to represent. Also, make sure your labels are appropriate and make sense with the question. A scale of Terrible to Okay isn't balanced, because the positive rating isn't strong enough. Also, a scale of Poor to Excellent doesn't make sense if the question is "How likely are you to recommend this product?"
- ✓ **Don't pre-fill the answers.** Don't start your survey with options or ratings already selected. Testers will be more likely to leave the question with the pre-filled answer, which could lead to inaccurate results.

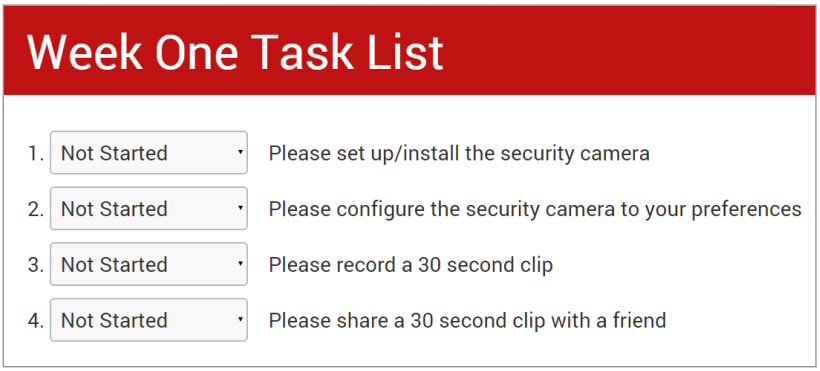
Tasks

Another important form of directed feedback is tasks. Tasks are specific activities you can assign your testers to perform and report back about. For example, it's common for beta teams to provide testers a list of tasks to get them started, such as installing the product and completing the onboarding process. You can also create tasks during your beta test asking testers to update to a newer version of your app or use specific features. You can have them test the range of your product in their home or the reliability when using it in different scenarios.

After your testers complete each task, they can report back on whether they were successful. You can then trigger follow-up questions accordingly. You can ask testers to report if they were unable to complete a task. You can use follow-up surveys to ask for more specific sentiments about the experience.

Tasks have a wide variety of use cases, which makes them a valuable part of the beta toolbox. You can use them to achieve just about any objective that requires testers interact with your product in a specific way. Keep this tool in your pocket throughout your beta test to help encourage participation and complete even the most specific goals.

As with surveys, it can be tempting to assign a lot of tasks to testers to get feedback on exactly the features you're interested in, but in doing so you lose valuable information on the natural user experience with your product. Make sure you balance this method with other forms of feedback to create a well-rounded beta experience for your testers.



Weekly task lists provide testers with some structure while still allowing plenty of opportunity to explore the product on their own.

Task Best Practices

Assigned tasks can serve a variety of important roles during beta testing, depending on your goals. Here's our advice on getting the most out of this method of feedback collection.

✓ **Give broad tasks to encourage early participation.** Some testers lack the initial drive to independently explore your product and report back their findings. We've found that giving people a set of very basic, general tasks will help kick-start their use of the product, after which they're more likely to do their own exploration. These should not include tasks that will focus the tester on very specific features or activities, but rather the product as a whole (e.g., download the software, load the software, review the online help documentation). In most cases, while you may have to nurture participation in the beginning, testers will be much more independent once they build some momentum.

✓ **Assign objectives rather than steps.** Rather than telling testers what to do step-by-step, give them a goal. This will better assess the product's usability. If you give them a task like "Change your avatar" you not only assess how the avatar process works, but also how easy or difficult it is to find and use it in your product.

✓ **Use tasks to gauge frequency.** Tasks are a great way to gauge how often an issue is occurring. You can assign a task to your testers to complete a certain action and see how many run into the issue. This will give you an idea of how widespread it is and if it's only affecting certain segments of your users.

✓ **Use specific tasks to regress fixes.** One area where a diverse and reliable tester team really shines is during regression testing. If you've fixed some known issues, verify you've solved the problem with a group (or, in some cases, all) of your testers. You can segment your team by test platforms that were known to exhibit the issue and assign tasks that follow the specific steps required to recreate it. Or, you can set your entire team after the problem just to make sure it's really gone. The added benefit of this is that testers will experience the results of their efforts firsthand, leading to increased participation.

✓ **Set deadlines, but make them reasonable.** It's important to attach deadlines to your tasks so testers feel a sense of urgency and don't let them languish. That said, make sure the deadlines are reasonable. We find that 2-3 days is a good standard for relatively simple tasks, while a week is appropriate for more complex assignments. You can opt for shorter deadlines when necessary (and only sparingly), but understand completion rates will suffer.

✓ **Time tasks to encourage participation.** If you're running a long test, you can use tasks to boost participation if testers start to drag. Giving them new things to do can inspire them to use the product in new ways, which will encourage additional ongoing feedback as well.

Additional Types of Directed Feedback

While the methods listed earlier are the most common types of directed feedback, there's a wide variety of activities you can use to achieve your goals. To give you an idea, here is a list of other forms of directed feedback we've seen work well:



Tester Calls

Conference calls (either one-on-one or with a group of testers) offer direct real-time communication with testers, similar to a focus group. These can be scheduled either early or late in a beta test, offering the product team the chance to talk directly with customers prior to release. These calls also increase participation rates by demonstrating the high value the company puts on beta testers and their feedback.



Site Visits

Visiting a beta tester is a great way to gain a first-hand understanding of the customer experience. Beyond the natural benefits of a face-to-face conversation, tester visits allow product teams to watch target customers perform tasks in their natural environments, providing valuable insight into real-world usage. Similar to tester calls, site visits can increase participation by making testers feel more connected to the beta project.



Videos

Requesting that testers submit videos of themselves using the product can provide valuable insight, similar to a site visit. You can ask testers to submit videos of specific activities (such as unboxing the product) or request video testimonials.



Directed Usage

In some cases a product team might not want feedback at all. Instead of wanting to know what testers think about their product, what they really want is more backend data that's generated by tester use. Asking testers to do certain tasks in certain ways or at certain times can provide important information about how your product performs in those scenarios, without testers saying a word.

There may be other assigned activities you'd like your testers to complete as part of your beta test. The flexibility of beta testing allows you to use many different tools to collect the right data to achieve your goals. Hopefully this has given you an idea of some of the tools at your disposal so you can leverage them during your next test.

Managing Directed Feedback

When it comes to managing directed feedback, your goal is to make sure all of your testers complete their activities so your data gives you as complete of a picture as possible. This involves implementing strategic tester compliance processes during your test and then reporting on the data appropriately once the activities are complete.

Tester Compliance

When employing directed feedback methods, it's important to get responses from all of your testers. If even a small number of your testers don't reply, it can affect your data in a big way. This reality is compounded even further when taking into account low participation rates that often accompany beta tests.

It's extremely important you not only have a plan for maximizing tester compliance, but you are also willing to put in the leg work it often takes to get high response rates.

Intro Calls

Depending on the size of your test, you should considering doing intro calls with each of your testers before your test begins. This allows testers to put a voice to a name and builds rapport. It's also a great opportunity to explain key aspects of your beta test, such as the nondisclosure agreement, the test schedule, and your participation expectations. Finally, it gives your testers a chance to ask any questions they might have before your test begins. This ensures that your testers are on the same page as your team from day one, which can have a huge impact on tester responsiveness and overall compliance.

Here are a few steps you can take to encourage compliance:

1. Before your test begins, establish participation expectations with your testers so they know what's expected of them. This can take a couple forms, including conducting intro calls, having testers sign a beta participant agreement, or providing detailed resources for your testers on how they can participate in your test.
2. Once your activities are posted, be sure to notify your testers so they can get started. In your notification, include the deadline for that activity to be finished. We assign activities on Wednesday and give our testers five days to complete most directed feedback. This ensures that they have the weekend to complete the requested tasks and surveys.
3. A few days before the deadline, send a gentle email reminder to let testers know the deadline is nearing.
4. Once the deadline passes, send another email reminding your tester to complete their activities. Remind them of the consequences of not participating in a timely manner (such as losing their opportunity for the project incentive or future testing opportunities).
5. If the tester still doesn't complete their assigned activities, try calling them to find out what is hampering their participation.

It can be helpful to have a team of pre-profiled alternates ready to jump in if you have to replace a sub-par tester. You can even start your test with a handful of extra testers, knowing that you may need to use them to bolster your participation numbers at some point.

Segmentations in Reporting

During recruiting you'll ask testers for key demographic and technical information to determine whether they're members of your target market. Make sure to hold onto that information so you can use it for reporting purposes throughout your test. While you're analyzing your results, it's helpful to be able to drill into your data based on these traits. That way you can compare installation experiences for iOS and Android users, or see if women gave your product better reviews than men. Having this information connected to their feedback gives your data much more depth. Beta management platforms like ours allow you to carry over data from your recruitment surveys into your project, but even if you aren't using a beta management platform with that functionality you can connect this information in a spreadsheet with a little extra effort.

Disseminating Your Data

All this data you've collected is only valuable if you get it into the hands of the people who can use it. Before you assign activities to your testers, think about which person on your team needs that data and what format would be most valuable for them. Set up as many reports as you can beforehand — that way you'll have a starting place once your data starts coming in.

It's also important to give context to your data whenever possible, especially when you're giving it to colleagues outside of your beta program. A product rating of three stars might not sound good, but if your industry average or your own company's historical score is two stars, then three stars is an impressive improvement.

Your context shouldn't just be quantitative, but qualitative as well. If 60 percent of your testers failed to install your app, provide some context in your report. Explain that this was the result of a new issue, which the testers helped you find and fix. Or maybe you worked with your testers to discover that the app installation process wasn't intuitive and have adjusted accordingly.

Getting the right data into the right hands at your organization is only part of the puzzle. You need to make sure they also have the appropriate context and analysis to use that data to make good decisions about the product.

Reactive Feedback

You can't plan for everything. In most beta tests some new objective or problem pops up that requires attention. As a result, we build some extra room into our beta tests for what we call reactive feedback. This allows us to pivot or add new objectives in the middle of a test so we can address the new issue.

For example, if you're testing a piece of software and discover a part of your installation process that's confusing and derailing half of your testers, you'll need to switch your focus to resolve the issue. You could develop a survey to get clarification on exactly where the confusion lies and how widespread it is. You could then use tasks to have testers walk through your revised process and give feedback on different steps. These activities will take time that would have otherwise been devoted to testing other parts of your product. As a result, it's important that you leave space for reactive feedback, so you can add activities as needed.

There are a few things to keep in mind when it comes to reactive feedback. First, you need to make sure you have the right testers to provide the feedback. If the uncovered issue only affects Windows Phones and you only have five testers with that phone in your test, you'll need to recruit additional testers to thoroughly scope and fix the issue. Second, make sure you aren't asking testers to do activities they aren't prepared for or are incapable of doing. If you decide halfway through your test that you need testers to record videos of themselves interacting with the product, some testers may not have the equipment or skills to do so. In these situations, you should consider running another phase of your beta test so you can recruit the right testers for the task at hand.

THE LONG TERM VALUE OF GOOD FEEDBACK PROCESSES

Building efficient and effective feedback processes can have a long term effect on your beta program. First, it improves the reproducibility of your beta tests. The next time you need to run a beta test you won't be starting from scratch. Instead, you'll already have your previous experiences and lessons learned to build on. You'll have templates to tweak and processes to strengthen. You'll have a bank of survey questions you can return to when you're designing your new surveys. This will save you valuable time and energy when your next beta test comes around.

Second, good feedback collection and management practices will give your beta program consistency. They'll create a consistent experience for your testers, who'll know what to expect and how to submit their feedback in future beta tests. It'll create consistent metrics for your product and quality managers to depend on each time they run a project. They'll also create consistent key metrics for your company's executives, who will be able to compare your company's products to each other, as well as a single product's changes over time. This will make your beta program more valuable and impactful across your organization.

CONCLUSION

Collecting high-quality beta feedback is about far more than just putting up a generic feedback form. You need to start with strategic objectives and then determine which feedback mechanisms from the beta toolbox work best to reach those objectives.

We hope that this whitepaper has helped you understand the ins and outs of feedback collection and how to use both ongoing and directed feedback to achieve your goals. Beta testing can have a huge impact on the success of your product, but it all relies on collecting high-quality feedback and then using it appropriately. If you can achieve that, then your beta program will become the rockstar of your product development life cycle.

centercode

Delta Testing Solutions

Leading product ops teams use Centercode to identify real-world issues to fix, areas to improve, and what their users love in order to build amazing products.



Centercode's delta testing platform is built on decades of experience user testing the world's most successful products. Our platform equips you with the automation, best practices, and processes you need to get actionable user insights for any use case.

PRODUCT LAUNCH

Engage your **Target Market** to successfully bring new products and services to market

USE CASES:

- Beta Tests
- User Acceptance Tests
- Field Tests

PRODUCT GROWTH

Recruit **Existing Customers** to continuously mature live products and services

USE CASES:

- Progressive Delivery
- Early Access
- Customer Advisory Boards

CUSTOMER ZERO

Gather **Employee** feedback to enhance your products while fostering a product culture

USE CASES:

- Alpha Tests
- Dogfooding
- Employee Tests

COMPETITIVE EDGE

Enlist **Competitors' Customers** to rapidly gain actionable intel about their products

USE CASES:

- Competitive Intelligence
- Market Testing
- Comparative Research

DELTA TESTING PLATFORM

Get a live demo of Centercode to experience the easy way to run your user tests, including:

- Auto-generated test designs
- Ted, your tester engagement bot
- Smart feedback prioritization
- Instant KPIs and dashboards

SCHEDULE A DEMO

WHY CENTERCODE?

Increase Revenue
½ ☆ INCREASE
in product ratings

Release Sooner
50% FASTER
test cycles

Gain Participation
75% INCREASE
in tester engagement

Save Time
40% + DECREASE
in time managing tests