



Best practices: from a new project to sign-off

The recommended order for a SurveyTester project: one language first, a manual run, test cases, SurveyBots, findings and issues, more languages, milestones, sign-off.

Edition October 2026

Online version with screenshots and links to the detailed articles:

<https://www.surveytester.com/academy/en/general-queries/project-best-practices/>

SurveyTester is a product of 2x4 Solutions GmbH · www.surveytester.com

Testing a survey well is less about tools than about order. Teams that start with ten languages, a hundred SurveyBots and no test cases produce a flood of issues nobody can prioritise. Teams that follow the sequence below find the structural problems first, with a single manual run, and let the automation do the repetitive work afterwards. This article is that sequence: eleven steps from an empty project to a signed-off survey, with the reason behind each step and the place in SurveyTester where it happens.

The sequence at a glance

Step	What you do	Where in SurveyTester
0	Collect the survey link, the questionnaire, the scope and the team	Outside SurveyTester
1	Create the project with one language	≡ project menu › Create New Project
2	Configure end pages, devices, permissions, sample, first milestone	Project Details › General, Screenshots, Permissions, Milestones
3	Test the main language by hand, and keep testing by hand alongside the bots	Test Survey › Manual Testing
4	Define test cases for completes and screenouts	Project Details › Test Cases, Questionnaire
5	Run SurveyBots	Test Survey › SurveyBot Testing
6	Accept or decline the automatic findings	Issues › findings
7	Review issues, routing and coverage	Issues, Pages & Routing
8	Add languages and markets, check translations	Project Details › Languages, Markets; Translations
9	Retest every script update against a milestone	Project Details › Milestones; Test Runs › Repeat
10	Sign off and wrap up	Project Details › Test plan, Insights, Lifecycle

Step 0: Before you start

A project goes faster when four things are on the table before anyone clicks **Create Project**:

- **The survey test link.** A link that opens the live, programmed survey. If the survey expects a respondent ID, the link can carry the placeholders {SAMPLEID} or {AUTONUMBER}; see step 2.
- **The latest questionnaire document.** The approved Word file is the specification every test is measured against. SurveyTester can read it (step 4), and testers need it as a reference anyway.
- **The scope.** Which languages, which markets, which device classes (desktop, phone, tablet) the survey will run in. Writing this list down is the first item of the default test plan, because every later step multiplies by it.

- **The team.** Who tests, who fixes, who decides when an issue is closed. Testers who only report need a different role from survey authors who fix and close.

Every tester needs the SurveyTester browser extension in Chrome or Edge: [Installing the SurveyTester Extension](#).

Step 1: Create the project with one language

Open the project menu (≡) in the top bar and choose **Create New Project**. The dialog is a single screen:

1. **Project Folder.** Most users work in one company, so choosing the folder is simply choosing where the project lives; the folder also decides which templates are offered. Everything else stays disabled until a folder is chosen.
2. **Project Name.** Checked for uniqueness while you type.
3. **Project Template.** The project type. It brings the issue classification fields and the values that close an issue, but also the settings behind the scenes: which automatic checks run and what they report, the end-page identifiers, the market and language questions. Your administrator maintains these; if there is only one, it is picked for you.
4. **Default language.** The language you will test first.
5. **Survey Link.** Paste the test link. SurveyTester recognises the survey platform from it and fills in **Survey System**; correct it if the dialog warns that the system does not match. If you do not know which platform the survey runs on, choose **Automatic**: SurveyTester then tries its platform-specific readers first and falls back to the generic one.
6. **Project Roles.** Which roles have access to the project. You can refine users and roles later under **Permissions**.
7. **Result filters.** Appears when result-filter templates exist for this folder and project template: your own saved filters or ones other users have shared. Your choice is remembered for the next project.
8. **Questionnaire (optional).** Drop the Word questionnaire here if you have one; the analysis runs in the background while you continue with step 2.

Tip: Start with one language, even for a 20-country study. Markets and further languages are added later under Project Details. The first manual run in the default language reveals the structural problems (routing, missing end pages, validation) that would otherwise have to be fixed and retested in twenty languages.

After **Create Project** the project opens on **Project Details > General**. For studies with one link per language, the **Create Multiple Projects** link at the bottom of the dialog accepts pasted text, for example the e-mail with all links, and works out languages and markets with AI.

Step 2: Configure the project

Spend ten minutes on **Project Details** before the first test. Each setting here saves work in every later step.

General

- **Survey End Pages.** Tell SurveyTester which pages mean *Complete*, *Screenout*, *Quota full* and *Early complete*, by keyword, redirect URL pattern or page name. Without this, every test run that stops is just "stopped" and test cases cannot be judged against their expected outcome. Administrators can define end pages once for the whole project type, so check what the project already inherited before adding your own. See [Project Level Survey End Pages](#).
- **Enable Auto Answer.** Lets SurveyBots and the manual tester's auto-answer button fill in questions. Keep it on.
- **Auto-Detect Issues** toggles. The automatic checks (spell check, visible HTML, unresolved piping, duplicate labels, missing images and more) run after every test run. Your administrator sets the defaults per company and project type; switch off here only what makes no sense for this survey. See [Auto Detect Issues](#).
- **Collect Accessibility Information** if WCAG matters for this survey.
- **Compare Project** if this is a wave of an existing study: SurveyTester then reports the differences in pages and questions.

Permissions. Add the users and roles who work on this project: [Add Role / User to a Project](#). External testers without an account get per-person links with an expiry date under **Test Links**.

Screenshots. Choose the devices SurveyTester should take screenshots on. Native screenshots from the extension are always there; the Screenshot Server adds Chrome, Firefox, iPhone, iPad and Android devices in both orientations. Pick what the scope from step 0 demands, not everything: every device multiplies the screenshots to review.

Sample. If the survey needs a respondent ID, click the `{SAMPLEID}` placeholder next to the Survey Link to open **Sample Management**, upload your sample records and map sample fields to questions. SurveyBots then type sample values where the survey asks for them. See [Using Sample with a project](#).

Milestones. Create the first milestone now, for example "Script v1". Milestones mark survey versions; every later comparison ("is the fix in?") is a comparison between milestones. See [How to Use Milestones](#).

Docs & Info. Upload the questionnaire and any briefing for testers, so everyone works from the same version.

Step 3: Test the main language once by hand

The first run should be a person, not a bot. Click **Test Survey** in the top bar, choose **Manual Testing** ("I will test the survey manually"), pick the language and **Desktop** or a **Mobile Device**, and press **Start Testing**. The browser extension opens the survey next to its own panel.

Walk through the survey as a respondent would and watch for:

- the welcome and consent pages, and whether the survey can be started at all with your link;
- every page's question text against the questionnaire;
- forced answers and validation messages: leave a page empty once, type letters into a numeric field once;

- the end page you reach, and whether SurveyTester recognised it as Complete (the run outcome in **Test Runs**).

Report everything you notice from the extension: it freezes the page, takes the screenshot and lets you mark the spot. See [Creating A New Issue](#).

Manual testing does not stop after this first run. Most teams keep testing by hand throughout the project: a tester picks a test case, a language and a device in the **Test Survey** dialog, walks through the survey and reports what they see from the extension. The SurveyBots cover volume and repetition; a person covers what no check can judge: whether a question makes sense, whether the layout works on a phone, whether an image fits, whether the tone matches the brand. Both kinds of runs appear in **Test Runs**, and issues from people and findings from the checks end up in the same list.

Tip: This first run is also how SurveyTester learns the survey. Every page, question and answer option seen in a test run appears in **Pages & Routing**, so after one manual run you can already review the page table, see which end page was reached and build test cases from real question names.

Step 4: Define test cases

A test case tells a tester or a SurveyBot how to answer specific questions and which end page is expected. Without test cases, SurveyBots answer at random, which exercises the survey but cannot prove that a quota cell is reachable or that a screener terminates where it should.

Open **Project Details > Test Cases**. The minimum set for almost every survey:

- **One complete per quota cell or routing branch.** Each fixes the answers that route into that branch and expects the outcome *Completed*.
- **One screenout per termination rule.** Each gives exactly one disqualifying answer and expects *Screenout*.
- **A random-answer complete** that leaves most questions free, for volume.

Three ways to build them, fastest first:

1. **From the questionnaire document.** If you uploaded the Word questionnaire, **Project Details > Questionnaire** analyses it with AI: questions, codes, routing, termination rules and quota cells. From that it proposes one complete per quota cell or branch and one screenout per termination rule, matches the document to the live survey and reports missing options, wrong exclusive flags and wording differences as findings.
2. **With AI from a description.** Write the respondent in plain language ("female, 35 to 44, owns a dog, buys premium food") and let **Generate with AI** turn it into a test case. **Generate AI Screenouts** then derives one screenout variant per qualifying constraint. See [AI-Powered Test Case and Response Generation](#).
3. **By hand.** Exact answers, percentage patterns, numeric ranges such as 18. .65, answers reused from an earlier question, and answers that depend on a sample field or an earlier answer. See [Create Test Case with additional Parameters](#) and [End Page Overview in Test Cases](#).

Tip: Name test cases after what they prove, not after who wrote them: "Complete: Dog owner, premium" and "Screenout: Age under 18" read well in the test-run grid, in result filters and in the sign-off report.

Step 5: Run SurveyBots

Now scale up. **Test Survey > SurveyBot Testing** ("SurveyTester creates automatic Tests") starts a job:

- **Languages, test cases and markets** to combine. Tick several of each; the job runs every combination.
- **Number of test runs**, in total or **per combination**. A good first batch is 3 to 5 runs per combination: enough to see the rotation and the routing, small enough to review the same day. Scale up to the hundreds for randomisation, MaxDiff designs and quota logic once the first batch is clean.
- **Answer mode**. *Standard* answers from the test case plus random choices, *Simple Text* for surveys with many open ends, *AI Standard* for realistic open-text answers, *AI with Persona* for demographics, dropout and inattention rates and realistic delays. See [Use Personas for Generating Answers](#).
- **Devices for Tests**. *Desktop*, or *Mobile* with an emulated phone or tablet and orientation. See [Test surveys on mobile devices](#).
- **VPN**. Let the bots enter the survey from the target country when the survey filters by location. See [SurveyBot Location](#).
- **Notification when the job is finished**. Jobs with hundreds of runs take a while; let SurveyTester mail you.

Each finished run appears in **Test Runs** with an outcome. Read them like this:

Outcome	Meaning	What to do
Complete, Screenout, Quota, Early complete	The run reached a configured end page	Compare with the test case's expected outcome
[Stopped]	The run ended on a normal page	The bot could not continue: validation it could not pass, a modal, a loop. Open the last step
[Error]	The survey showed an error page or the page could not be read	Usually a survey-side error; report it
Waiting for input	The bot met a CAPTCHA	A person solves it in the extension; the bot continues

Tip: Bots do not get tired, but reviewers do. Start small, fix what the first batch finds, then run the big batch. SurveyTester reports a recurring problem once, not once per run, but a thousand runs against a broken screener still stop at the same page and prove nothing about the rest of the survey.

Step 6: Let the automatic checks work

After each test run the automatic checks examine every page: spelling against the company dictionary, text in the wrong language, visible HTML, unresolved piping such as ``${q1}`` or `[pipe:q1]`, empty or duplicate labels, inconsistent numeric ranges, missing images, missing rotation, answer options that should depend on an earlier answer, MaxDiff balance, and personal data on the page. The results wait as **findings** on the **Issues** tab.

Review them in one sitting after the first bot batch:

- **Accept and Create Issue** turns a finding into a normal issue with screenshot and classification.
- **Decline Finding** with a reason removes it. Declining a spelling finding as "brand name" or "correct in this language" can add the word to the company dictionary, so the same word is never reported again. See [Spell check](#) and [Company Dictionary](#).
- Findings of the cross-run checks (rotation, item dependency, MaxDiff balance) are withdrawn on their own once new runs no longer show the deviation.

Two checks deserve a closer look on surveys that use them: the rotation and item-dependency checks, which judge across many runs whether a list was really randomised and whether a later question showed only the items chosen earlier, and the MaxDiff balance check described in [Testing MaxDiff questions](#). Both need volume: that is the second reason for the larger bot batch in step 5.

Step 7: Review issues, routing and coverage

Issues. Every issue carries the classification fields from your project template (for example severity, area, status) and a phase. Use the phases as the hand-over protocol between testers and survey authors: *Initial* while testing, *Ready for QC* when the author has fixed a batch, *Script Update Needed* when the test confirms the problem. Mention people with @ in comments; SurveyTester notifies them by e-mail according to their subscription settings. See [Email notifications and relevant project phases](#).

Result filters narrow every project screen to one slice: a language, a test case, the runs since a milestone, open issues of one severity. Save the slices you use daily as templates. See [Result Filter](#).

Pages & Routing. The **Routing Diagram** shows every page as a node with the number of answers and the open issues on it. Paths that no run has taken are the gaps in your coverage; **Path Analysis** counts the distinct routes and how often each was tested, see [Survey Path Analysis](#). For Dimensions and Qualtrics surveys, **Upload Survey Metadata** (`.mdd` or `.qsf`) adds the questions no run has reached yet, so coverage gaps become visible; see [Survey Metadata Upload](#).

Tip: Review in this order: findings, then open issues sorted by severity, then the routing diagram for untested branches. Each pass answers a different question: "is the text right?", "is the logic right?", "did we test everything?".

Step 8: Add languages and markets, check translations

Once the default language is stable, add the rest under **Project Details > Languages** and **Markets**: [Adding Multiple Languages](#), [Adding Multiple Markets](#), and for the combinations of both [Multi-Country, Multi-Audience Project Setup](#).

Then repeat the proven test cases per language with a SurveyBot job (tick all languages, the same test cases) and open the **Translations** tab:

- The word-level comparison shows every label side by side across languages, test runs and milestones.
- **Show Identical** lists labels that are the same in two languages, which is the fastest way to find text that was never translated.
- A Word translation file can be compared against the survey, and differences become issues.

The language and market questions of the survey can be mapped in the project template so that SurveyTester knows which run belongs to which market even when the link does not say so.

Step 9: Retest every script update against a milestone

Testing is a loop: test, report, fix, retest. SurveyTester keeps the loop measurable with two tools.

Milestones. Each time the survey author publishes a new script version, add a milestone ("Script v2") before the first new test run. Result filters, the Translations tab and the findings then let you compare "since v2" with "before v2", and the phase statistics in Insights count the rounds. The recommended cycle is described in [Project Phase & Milestone](#).

Repeat Test Run. From any test run, **Repeat** replays the same answers, from any page, in another language or on another device, and compares the repeated run with the original showing only the differences. That is the quickest proof that a fix is in. See [Repeat Test Run in SurveyTester](#).

External reviewers, for example the end client, can join at this point through **Test Links**: a personal link with an expiry date, no account needed, with the right to add issues.

Tip: Never let the author update the script between a test batch and its review without a milestone in between. Without the marker you cannot tell which issues were seen on which version.

Step 10: Sign off and wrap up

Test plan. **Project Details > Test plan** is the project's checklist. **Load template** fills it with your company's template or the SurveyTester default of 84 items in 13 phases, from "current questionnaire is available to testers" to "sign-off recorded". Items can be ticked per language, market or device, and about a third of them tick themselves from the evidence in the project: a questionnaire was uploaded, every test case reached its expected end page, the accessibility report is green. **Refresh automatic checks** re-evaluates them. Open items are the agenda for the last review meeting.

Insights. Project Details > Insights summarises the project: manual against bot runs, issues found by people against automatically, open against resolved, rounds between QA and authors, tester time, the estimated length of interview per test case, and the project status checklist. Each panel carries the review question it answers.

Exports for the hand-over:

- Issues as Excel from the Issues download center, see [Exporting Issues to Excel](#);
- all screenshots as ZIP, filtered by language, market, test case or milestone, see [Export Screenshots](#);
- the Word questionnaire as tested, per language, from the **Pages & Routing Download Center**, see [Questionnaire export](#);
- the Quality Management Report of a single test run from the Test Runs download center.

The last word is human. The checks, the bots and the test plan narrow the work down, but not every problem is found automatically, and no finding closes itself. Someone reads the open items, decides what is a defect, what is accepted as is and what is out of scope, and records that decision in the issue. Sign-off is that person's signature, not the automation's.

Sign-off and close. Project Details > Lifecycle offers **Project Sign-off**, which notifies every project user and everyone with the sign-off permission; it warns when issues are still open. Then create a backup, set the project to not active so nobody tests a signed-off script by accident, and keep it for the next wave as a **Compare Project**.

Common mistakes

- **All languages on day one.** Twenty copies of every structural issue. Stabilise one language first.
- **No end pages configured.** Every run ends as "stopped", test cases cannot be judged, Insights stay empty.
- **Bots without test cases.** Random answers cannot prove a quota cell or a termination rule.
- **Declining findings without a reason.** The same spelling finding returns on the next run; a reason teaches the dictionary.
- **Script updates without a milestone.** Nobody can say which version an issue was seen on.
- **Reviewing a thousand runs at once.** Run a small batch, fix, then scale.
- **Testing on every device from the start.** Pick the devices from the scope; add the rest for the final round.
- **Relying on the automation alone.** The checks find what can be counted; a person finds what makes no sense.

Key takeaway

One language, one manual run, then test cases, then bots, then everything else. SurveyTester's automation pays off once the structure is right, and the structure is found fastest by a person walking through the main language once. Everything after that is a loop of test, fix and retest, by people and by

bots side by side, that milestones keep measurable and the test plan brings to a defensible end. The automation narrows the work; a person makes the final call.