Quathera

User Guide

Complete guide for using Quathera's AI-powered form testing platform. Covers installation, configuration, form discovery, test execution, and results analysis.

Version 1.0 | 2025

Table of Contents

1. Introduction

Quathera is an AI-powered platform that automatically discovers, maps, and tests web forms in your applications. Unlike traditional test automation that requires scripting, Quathera uses artificial intelligence to:

- Discover all form pages in your web application automatically
- Map every field, understand conditional logic, and detect relationships
- Generate complete test scenarios (Create, Verify, Edit, Verify cycles)
- Execute tests and self-heal when UI changes break locators
- Report bugs with screenshots and actual vs. expected comparisons

Key Benefits:

- No coding or scripting required
- Tests adapt automatically when your UI changes
- Complete test coverage including negative tests
- Your credentials never leave your network (agent runs locally)

2. Installing the Desktop Agent

The Quathera Desktop Agent runs on your local machine and executes all browser automation. This ensures your test credentials and application data never leave your network.

Downloading the Agent:

1. Log in to the Quathera web application

2. Click 'Download Agent' button in the header

3. Select your operating system: Windows, macOS, or Linux

4. Download the installer

Installation Steps:

Windows:

- Double-click the .exe installer
- Follow the installation wizard
- Launch 'Quathera Agent' from Start Menu

macOS:

- Open the .dmg file
- Drag Quathera Agent to Applications
- Launch from Applications folder
- Grant permissions when prompted (Accessibility, Screen Recording)

Linux:

- AppImage: chmod +x QuatheraAgent.AppImage && ./QuatheraAgent.AppImage
- Debian/Ubuntu: sudo dpkg -i quathera-agent.deb

Connecting the Agent:

1. When you launch the agent, it displays a connection code

2. In the web application, you'll be prompted to enter this code

3. Once connected, the header shows 'Agent Online' (green)

4. The agent must remain running during discovery and test execution

3. Creating Projects

Projects organize your testing by web application. Each project has its own test sites, discovered forms, and test results.

Creating a New Project:

1. Click 'Create Project' button in the header or dashboard
2. Enter a project name (e.g., 'HR System - QA', 'E-Commerce Portal')
3. Optionally add a description
4. Click 'Create'

Switching Between Projects:

Use the project dropdown in the header to switch between projects. All test sites, discovered forms, and results are specific to the selected project.

4. Configuring Test Sites

Test Sites define WHERE to test and WHO to test as. Different user types (Admin, Regular User, Manager) often see different forms, so you'll create a test site for each user type.

Understanding Test Sites:

Your web application likely has different user roles:

- Admin users see admin panels, user management, settings
- Regular users see customer-facing forms
- Managers might see approval workflows

Each role may see different forms, so create a test site for each.

Environments:

You can also test across multiple environments:

- QA: Where AI discovery happens (safest for exploration)
- Staging: Pre-production validation
- Production: Smoke tests, monitoring

Adding a Test Site:

1. Go to your project dashboard
2. Click 'Test Sites' in the sidebar
3. Click 'Add Test Site'
4. Fill in the details:
    - Name: Descriptive name (e.g., 'Admin - QA')
    - URL: The login page URL (or starting page if no login required)
    - Environment: QA, Staging, or Production
    - Requires Login: Yes or No
    - Username & Password: If login is required
5. Click 'Save'

Login vs. No-Login Sites:

Login Sites: The agent will authenticate using provided credentials before discovering/testing.

No-Login Sites: For public pages, Electron apps, or kiosk modes where no authentication is needed.

5. AI Form Discovery

Form discovery is the first step. The AI automatically crawls your web application and finds every form page.

Starting Discovery:

1. Go to 'Form Pages Discovery' in the sidebar
2. Select which test sites to scan (check the boxes)
3. Click 'Start Discovery'
4. The agent launches a browser and begins crawling

What AI Does During Discovery:

- Logs in automatically using your test site credentials
- Navigates through your application like a real user
- Clicks on menus, buttons, and links to explore all pages
- Identifies pages that contain forms (data entry, registration, etc.)
- Records the navigation path to reach each form
- Handles SPAs (Single Page Applications), iframes, and dynamic content
- Performs visual verification to catch UI issues

Discovery Results:

After discovery completes, you'll see a list of discovered forms showing:

- Form Name: AI-generated name based on the form's purpose
- Path Steps: Number of navigation steps to reach the form
- Type: Root (standalone) or Child (depends on a parent form)
- Discovery Date: When the form was discovered

6. AI Form Mapping

After discovery, each form needs to be mapped. Mapping is where AI performs deep analysis to understand every aspect of the form.

Starting Mapping:

1. In the Discovered Forms list, find the form you want to map

2. Click the Map icon (■■)

3. AI automatically begins the mapping process

What AI Discovers During Mapping:

All Form Fields:

- Text inputs, textareas, dropdowns, checkboxes, radio buttons
- Date pickers, file uploads, rich text editors
- Hidden fields and dynamically generated fields

Conditional Logic (Junctions):

- Fields that appear/disappear based on other field values
- Example: Selecting 'Country: USA' reveals 'State' dropdown
- AI maps ALL possible paths through the form

Form Relationships:

- Parent-child relationships (e.g., Customer → Order → Line Item)
- Which forms must be created before others

Multi-User Workflows:

- Which user types can access this form
- Required login/logout cycles for testing

Validation Rules:

- Required fields
- Format validation (email, phone, etc.)
- Error messages shown for invalid input

Spec vs. Discovery:

If you've uploaded a specification document, AI compares what it discovers against your spec and flags any discrepancies:

- Fields in spec but not found in the form
- Fields in form but not documented in spec
- Differences in field types or validation rules

7. AI Test Scenario Generation

Once forms are mapped, AI automatically generates comprehensive test scenarios.

What AI Generates:

Complete CRUD Test Cycles:

- CREATE: Fill the form with test data, submit
- VERIFY: Check all values were saved correctly (grid + detail view)
- EDIT: Modify the values
- VERIFY: Confirm changes were saved

Junction Path Variations:

- For forms with conditional fields, AI generates scenarios for each path
- Example: Full-Time employee form vs. Part-Time employee form

Negative Test Cases:

- Missing required fields
- Invalid data formats
- Boundary conditions

Multi-User Scenarios:

- Automatic login/logout cycles with appropriate users
- Testing access control (can user A see form B?)

End-to-End Workflows:

- For parent-child forms, AI creates scenarios that create parent first, then child
- Example: Create Department → Create Employee → Create Task

Viewing Generated Scenarios:

Go to 'Test Scenarios' in the sidebar to see all generated test cases. Each scenario shows:

- Scenario name and description
- Forms involved
- User types required

- Junction path (if applicable)

8. Running Tests

With scenarios generated, you can now execute tests. AI handles everything automatically, including self-healing when locators break.

Starting a Test Run:

1. Go to 'Run Tests' in the sidebar

2. Select which scenarios to run (or 'Select All')

3. Choose the environment: QA, Staging, or Production

4. Select browser: Chrome, Firefox, Edge, or Electron

5. Click 'Run Tests'

What Happens During Test Execution:

For each test scenario, AI:

- Launches the browser on your machine (via the Desktop Agent)
- Logs in with the appropriate user
- Navigates to the form using the discovered path
- Fills fields with test data
- Submits the form
- Verifies values were saved correctly
- Performs visual verification to catch UI issues
- Captures screenshots on any failure

Self-Healing:

When a step fails because a locator changed (element ID, class, etc.):

- AI analyzes the current page
- Finds the element using visual and structural analysis
- Automatically updates the locator
- Retries the step
- If successful, saves the corrected locator for future runs

Error Handling:

AI intelligently handles various errors:

- Locator Changed: Auto-corrects and continues
- Validation Error: Records the error and continues or retries
- Network Error: Captures server/frontend errors for debugging
- Page State Lost: Restarts the test from a clean state

9. Viewing Results & Logs

Test results and logs are available on both the Desktop Agent and the Web Application.

Dual View Logs:

Agent Logs: Real-time logs showing exactly what's happening during execution

Web App Logs: Same information, accessible from any browser

What Logs Show:

- Running scenarios and their progress
- Each step executed (fill, click, verify, etc.)
- Pass/fail status for each step
- Self-healing actions (locator corrections)
- Bugs found with actual vs. expected values
- Network/sniffer errors from server or frontend
- Visual verification issues detected
- Screenshots at failure points

Screenshots:

Screenshots are captured automatically when:

- A test step fails
- A bug is detected
- Visual verification finds an issue

Screenshots are attached to the bug report and available in the logs.

10. Test Coverage Matrix

The Test Coverage Matrix provides a visual overview of test results across versions.

Understanding the Matrix:

- Columns: Application versions tested
- Rows: Test scenarios
- Cells: Color-coded results
  - Green: 100% of tests passed
  - Orange: Some tests passed (50-99%)
  - Red: Many tests failed (<50%)

Using the Matrix:

- Quickly see which versions have problems
- Identify which scenarios are flaky
- Track quality trends over time
- Export reports for stakeholders

11. Bug Reports & Jira Integration

When tests find real bugs, Quathera automatically creates detailed bug reports.

Bug Report Contents:

- Test scenario name
- Step that failed
- Expected value vs. Actual value
- Screenshot at failure point
- Navigation path to reproduce
- Environment details (browser, URL)

Jira Integration:

Configure Jira in Project Settings to enable:

- Automatic bug creation when tests fail
- Duplicate detection (AI checks if similar bug exists)
- Regression identification (AI flags if a closed bug reappears)
- Full context attached to Jira tickets

12. Troubleshooting

Agent Not Connecting:

- Ensure the agent is running
- Check your internet connection
- Try restarting the agent
- Verify the connection code is entered correctly

Discovery Not Finding Forms:

- Check test site credentials are correct
- Ensure the starting URL is accessible
- Try increasing discovery depth in settings

Tests Failing Unexpectedly:

- Check if the application is up and accessible
- Review the logs for error details
- Verify test data is valid for the current app state

Need More Help?

- Email: support@quathera.com
- Documentation: quathera.com/docs