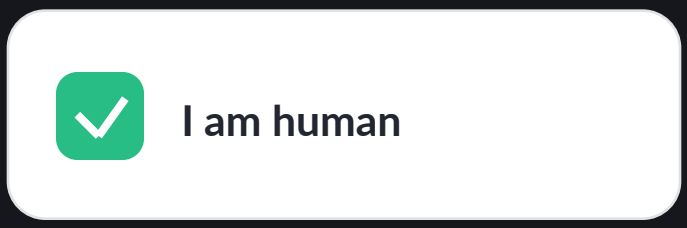
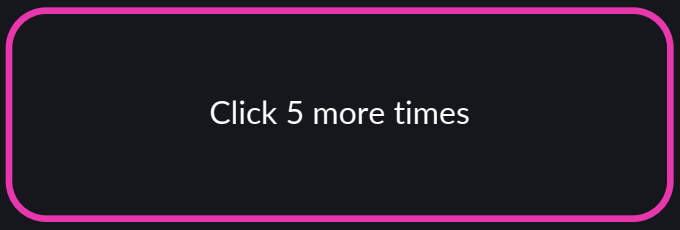
04 DEC 2025

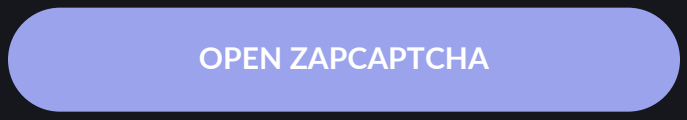
Zap Captcha

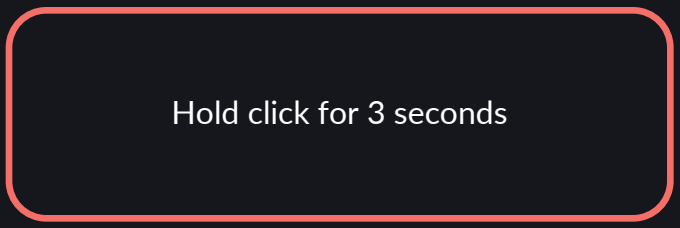
THE CHALLENGE PLAYBOOK

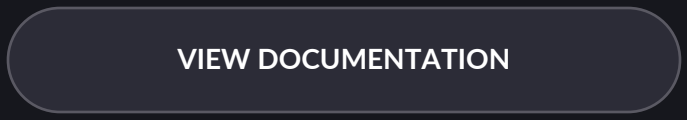
A complete technical guide to custom challenges, browser controls, and real-world verification workflows.











01 / FIELD NOTE

Find the right part, fast.

A full implementation manual, organized from first install through deployment decisions.

01 INTRODUCTION & QUICKSTART 03 - 05

02 FORM & ACTION INTEGRATIONS 06 - 08

03 FLAGS & SECURITY CONTROLS 09 - 12

04 USER EXPERIENCE MODES 13 - 15

05 INTEGRITY & SERVER MODE 16 - 18

06 ACCESSIBILITY & TROUBLESHOOTING 19 - 20

07 DEPLOYMENT & REFERENCE 21 - 22

SOURCE / QVLx ZAPCAPTCHA QUICKSTART & CONFIGURATION REFERENCE

02 / FIELD NOTE

Where verification belongs.

A challenge layer you can attach to a form or a custom application action.

A PROTECTED INTERACTION, END TO END

01 **LOAD** Module

02 **PLACE** Container

03 **VERIFY** Challenge

04 **RUN** Callback

Form submissions

Guard contact, feedback, account registration, and other form actions.

Custom actions

Run site-specific callbacks only after the challenge succeeds.

Multiple instances

Separate challenge containers can guard independently protected controls.

Configuration system

Fine-tune behavior using attributes and the global zap-flags meta tag.

DESIGN PRINCIPLE

Verify the interaction. Guard the underlying operation.

03 / FIELD NOTE

Start clean. Start small.

The standard setup loads the module, then pairs one CAPTCHA container with a protected button.

01 / MODULE SCRIPT

HTML / BEFORE </BODY>

```
<script type="module"
  src="https://zapcaptcha.com/zapcaptcha.js">
</script>
```

02 / TARGETED CONTROL

HTML / BASIC PAIR

```
<div class="zcaptcha-box"
  data-target-id="sendBtn"
  data-zcap-timeout="20"></div>
<button id="sendBtn" class="zapcaptcha-button"
  type="button" disabled>Send</button>
```

MOST IMPORTANT CONTRACT

The data-target-id value must match the id of the protected button. Keep the button disabled initially so ZapCaptcha controls the verification state.

04 / FIELD NOTE

A small DOM contract.

Classes, attributes and IDs connect each challenge to the correct action.

MARKUP	ROLE	NOTES
zcaptcha-box	Challenge host	Container class
data-target-id	Protected element ID	Must match target id
data-zcap-timeout	Timeout value	20 seconds documented default
data-size	Display mode	compact supported
data-zcap-frictionless	Opt-in on container	Use with frictionLess global flag
zapcaptcha-button	Target button class	Target initially disabled
disabled	Initial state	Handled by integration

INTEGRATION CHECK

Match IDs exactly. Avoid duplicates. Bind each control once, and preserve native form validation before executing callbacks.

05 / FIELD NOTE

Respect native validation.

Preserve required-field checks before a successful challenge permits submission.

HTML / FORM

```
<form id="feedback" action="/submit" method="POST">
  <input name="message" required>
  <div class="zcaptcha-box"
    data-target-id="send" data-zcap-timeout="20"></div>
  <button id="send" type="submit"
    class="zapcaptcha-button" disabled>Send</button>
</form>
```

JS / VERIFICATION HANDOFF

```
window.addEventListener("DOMContentLoaded", () => {
  const form = document.getElementById("feedback");
  const btn = document.getElementById("send");
  if (!form || !btn) return;
  btn.addEventListener("click", event => {
    event.preventDefault();
    if (!form.reportValidity()) return;
    window.ZapCaptcha?.verify(btn, () => form.submit());
  });
});
```

SERVER ENFORCEMENT

A success callback permits a browser action. Sensitive server endpoints need independent server-side verification.

06 / FIELD NOTE

Verify, then act.

Apply the same verification call to a protected JavaScript action—not only a form.

HTML / WIDGET

```
<div class="zcaptcha-box" data-target-id="verifyBtn"
  data-zcap-timeout="20"></div>
<button id="verifyBtn" type="button"
  class="zapcaptcha-button" disabled>Verify</button>
<div class="inline-result"></div>
```

JS / SUCCESS CALLBACK

```
const button = document.getElementById("verifyBtn");
button?.addEventListener("click", event => {
  event.preventDefault();
  window.ZapCaptcha?.verify(button, () => {
    document.querySelector(".inline-result")
      .textContent = "Human verified!";
  });
});
```

IMPORTANT DISTINCTION

A client-side challenge is an interaction gate. The browser callback alone is not server-verifiable proof.

07 / FIELD NOTE

One page. Many gates.

Give every protected control its own target identifier and callback.

HTML / TWO TARGETS

```
<div class="zcaptcha-box" data-target-id="saveBtn"></div>
<button id="saveBtn" class="zapcaptcha-button"
  disabled>Save</button>
<div class="zcaptcha-box" data-target-id="shareBtn"></div>
<button id="shareBtn" class="zapcaptcha-button"
  disabled>Share</button>
```

JS / BIND ONCE

```
document.querySelectorAll(".zcaptcha-box").forEach(box => {
  const id = box.getAttribute("data-target-id");
  const button = document.getElementById(id);
  if (!button || button.dataset.bound) return;
  button.dataset.bound = "true";
  button.addEventListener("click", e => {
    e.preventDefault();
    window.ZapCaptcha?.verify(button, () => {
      /* Perform the intended action here. */
    });
  });
});
```

WIRING RULE

Use per-control callbacks. If forms are involved, validate the correct form before starting verification.

08 / FIELD NOTE

One compact control plane.

Global settings live in zap-flags; combine documented flags and intentional overrides.

HTML / GLOBAL META CONFIGURATION

```
<meta name="zap-flags"
  content="allsec = true, lockoutsEnabled = false">
```

Presets and overrides

allsec activates the documented countermeasure collection. Additional values can override an option.

Intentional profiles

Choose controls for your traffic, compatibility needs and interaction model—not a universal preset.

A CURATED, INCREMENTAL PROFILE

HTML / EXAMPLE

```
<meta name="zap-flags"
  content="canvasMode = true, viewportConfined = true,
    lockoutsEnabled = true, debugMessages = false">
```

Flag names are case-sensitive in the published examples. Test combined configuration against your installed ZapCaptcha build.

09 / FIELD NOTE

Know your starting point.

Documented defaults for frequently exposed configuration settings.

FLAG	DEFAULT	PURPOSE
viewportConfined	true	Keep challenge within viewport
canvasMode	true	Use Canvas-backed presentation
lockoutsEnabled	true	Client-side lockout controls
sessionLock	true	Session-backed lockout state
cookieLock	true	Cookie-backed lockout state
localLock	false	Optional local-storage persistence
softLock	false	Disable action instead of hard lock
serverMode	false	Optional server integration mode
serverLock	false	Server-authoritative lockout when set up
clickDelay	false	Randomized action delay
spawnDelay	false	Randomized challenge appearance
frictionLess	false	Frictionless behavior support
debugMessages	false	Diagnostic message output

Defaults follow the published QVLx quickstart. Confirm behavior when changing releases.

10 / FIELD NOTE

Hardening beyond the puzzle.

Optional browser-side checks target modifications to expected resources and execution behavior.

FLAG	DOCUMENTED PURPOSE
checksumJS	Check expected JavaScript hash
checksumCSS	Check expected stylesheet hash
functionTamperCheck	Inspect selected core-function modifications
canvasSpoofingCheck	Inspect selected Canvas manipulation signals
cssOverrideDetection	Look for selected style overrides
clickBlockEnforcement	Enforce selected protected-click behavior
lockShadow	Apply Shadow DOM-related protection
monkeyPatching	Modify selected DOM creation functions

COMPATIBILITY FIRST

Integrity and DOM-interception checks can conflict with scripts, extensions, and legitimate overrides. Enable deliberately; evaluate in staging.

11 / FIELD NOTE

Understand the surroundings.

Optional signals can inspect the execution environment and widget embedding.

FLAG	DOCUMENTED PURPOSE
headlessBrowserCheck	Selected headless-browser indicators
iframeEmbedCheck	Whether the page is running in an iframe
iframeInjectCheck	Presence of selected injected iframe elements
torCheck	Selected Tor-related signals
vpnCheck	Selected VPN-related signals
devtoolsCheck	Selected developer-tools activity

Tune for legitimate users

Environment signals are imperfect indicators of automated abuse. Expect false positives.

Observe before restricting

Use testing or monitoring profiles to identify accessibility, privacy-tooling and browser-compatibility effects.

12 / FIELD NOTE

Make the challenge fit.

Responsive presentation depends on placement, timeout, and available space in the host interface.

Viewport confinement

viewportConfined = true is the documented default and helps keep the challenge visible.

Canvas presentation

canvasMode = true is the documented default for Canvas-backed rendering.

Compact size

Use data-size="compact" on the challenge container for narrow cards and mobile layouts.

Timeout control

Set data-zcap-timeout as needed. The quickstart documents 20 seconds by default.

HTML / COMPACT

```
<div class="zcaptcha-box"
  data-target-id="continueBtn"
  data-size="compact"
  data-zcap-timeout="120"></div>
```

Test touch access, zoom, small screens, viewport resizing and browser input methods.

13 / FIELD NOTE

Less friction, on purpose.

Frictionless behavior is opt-in and can be paired with a per-container attribute.

HTML / TWO-STEP ENABLEMENT

```
<meta name="zap-flags" content="frictionLess = true">
<div class="zcaptcha-box"
  data-target-id="submitBtn"
  data-zcap-frictionless="1"></div>
```

clickDelay

An optional randomized click delay; disabled by default.

spawnDelay

An optional randomized challenge appearance delay; disabled by default.

A GOOD USER EXPERIENCE PROFILE

Enable frictionless behavior deliberately. Keep a clear fallback and test randomized delays against accessibility requirements before deployment.

14 / FIELD NOTE

Choose how restrictions behave.

Browser storage and enforcement flags support different lockout behaviors.

FLAG	INTENT
lockoutsEnabled	Enable client-side lockouts
sessionLock	Use session storage for lockout state
cookieLock	Use cookies for lockout state
localLock	Persist state in local storage; off by default
softLock	Disable protected actions rather than hard lock
serverLock	Enable server-authoritative lockouts when configured

HTML / LOWER-PERSISTENCE STAGING PROFILE

```
<meta name="zap-flags"
  content="lockoutsEnabled = true, sessionLock = true,
  cookieLock = false, localLock = false">
```

Browser-side lockout state can be cleared or bypassed. Server-authoritative policy requires a separately configured server integration.

15 / FIELD NOTE

Expect the right bytes.

The published reference documents SHA-384 resource hashes and related validation flags.

HTML / <HEAD>

```
<meta name="zap-js-integrity"
  content="sha384-JS_HASH_HERE">
<meta name="zap-css-integrity"
  content="sha384-CSS_HASH_HERE">
<meta name="zap-flags"
  content="checksumJS = true, checksumCSS = true">
```

Compute from exact resources

Expected digests must match the deployed script and stylesheet bytes for the active release.

Re-check after updates

Whenever resource files change, verify and publish corresponding digests with the release.

DO NOT COPY PLACEHOLDERS

The example shows the documented format, not working SHA-384 values. Compute hashes from the exact files you serve.

16 / FIELD NOTE

Different enforcement boundaries.

A browser interaction gate and a backend authorization decision serve different purposes.

Browser-only operation

serverMode = false by default. The widget can gate buttons and trigger callbacks in the browser.

Server-integrated operation

serverMode = true requires separate server integration. Sensitive operations need server-side decisions.

THE TRUST PATH

01

BROWSER

Initiates challenge

›

02

APPLICATION

Handles action

›

03

BACKEND

Enforces operation

BOUNDARY RULE

A disabled button or local JavaScript callback cannot establish authorization for an independently callable server endpoint.

17 / FIELD NOTE

Server mode, clearly defined.

The public quickstart documents serverMode and serverLock, but not a universal verification protocol.

HTML / INTENT

```
<meta name="zap-flags"
  content="serverMode = true, serverLock = true">
```

DEFINE THESE IN YOUR SERVER INTEGRATION

01 The verification proof or server response contract supplied by the installed server integration.

02 The backend endpoint accepting the protected operation and checking its authoritative result.

03 Failure, expiration, replay and user-session behavior appropriate to the protected action.

04 Observability and test cases for success, rejection and network failure.

DOCUMENTATION SCOPE

These are integration-design considerations, not invented ZapCaptcha API endpoints. Use your official server integration reference for the request and response contract.

18 / FIELD NOTE

Verify without excluding.

Good testing covers the entire journey, not only the successful click path.

Keyboard and focus

Check predictable focus, challenge navigation and keyboard-only paths.

Mobile and zoom

Test portrait layouts, compact mode, touch targets, resizing and viewport fit.

Error recovery

Recover from timeout, rejection and interruption without losing unrelated form data.

Feedback states

Communicate pending, complete and failed states in understandable language.

QA MATRIX

CONTEXT	TEST
Desktop mouse	Normal challenge + callback
Keyboard only	Focusable, perceivable controls
Mobile / touch	Compact presentation, viewport fit
Screen magnification	Usable layout at zoom
Interrupted challenge	Recovery and retry workflow

19 / FIELD NOTE

Diagnose the failure stage.

Trace the protected action from script loading through binding, verification, and callback.

SYMPTOM	FIRST CHECKS
No CAPTCHA appears	Check module, JS, class, and CSP
Button stays disabled	Check initialization, target ID, and lockouts
Callback does not run	Check verify() binding and success
Form is not submitted	Check validity, callback, and form target
Mobile challenge is awkward	Try compact mode and viewport confinement
Unexpected lockouts	Review lockout storage and detection flags
Blocked by CSP	Allow required resources without broad exceptions

BEST DEBUGGING SEQUENCE

Script loaded → host detected → target matched → challenge reachable → callback invoked → action completed.

20 / FIELD NOTE

Match the site, not a preset.

Use documented settings as starting profiles, then validate against your actual application.

01 STANDARD FORM

Conventional visible challenge for contact, feedback, or registration forms.

```
viewportConfined = true, canvasMode = true
```

02 COMPACT / MOBILE

Narrow containers; pair with data-size="compact" on the widget.

```
viewportConfined = true, canvasMode = true
```

03 TARGETED HARDENING

Incremental protection flags with compatibility and false-positive review.

```
clickBlockEnforcement = true, canvasSpoofingCheck = true
```

DEPLOYMENT BOUNDARY

Server mode requires separately configured server enforcement and trust design.

Ready for the human moment.

A practical final checklist for a reliable verification experience.

- [x] 01 The module loads and the widget initializes.
- [x] 02 Every target ID and button association is correct.
- [x] 03 Form validation runs before verification.
- [x] 04 Mobile, keyboard and recovery paths are tested.
- [x] 05 Configuration flags behave as intended.
- [x] 06 Resource hashes match the deployed build.
- [x] 07 Sensitive server actions are enforced server-side.
- [x] 08 Launch and failure scenarios are documented.

OPEN ZAPCAPTCHA | QUICKSTART GUIDE | CONTACT QVLx

EXTENDED KNOWLEDGE BASE

zapcaptcha.com/docs