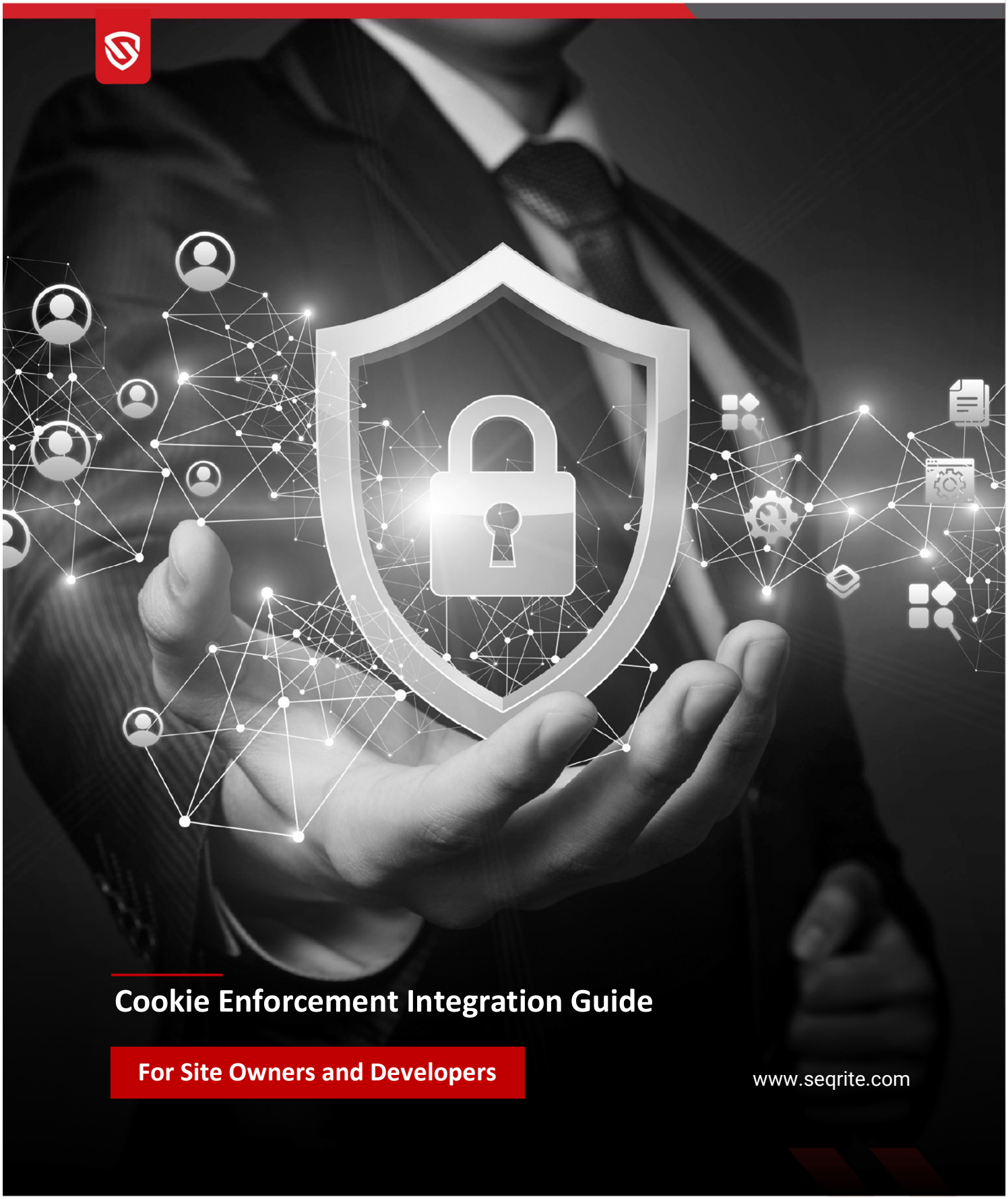
SEQRITE



Cookie Enforcement Integration Guide

For Site Owners and Developers

www.seqrite.com

Contents

Overview 2

What will you receive? ***2***

How does enforcement work? ***2***

Applicable Steps Based on Your Setup/ Follow the Steps That Apply to You 2

Step 1: Embed the Four Scripts 3

Step 2: Gate Your Tracking Scripts 3

Facebook / Meta Pixel ***4***

HotJar ***4***

Generic inline script ***5***

Step 3: Block the GTM Container (GTM Sites Only) 6

Step 4: Configure GTM Custom Event Triggers (GTM Sites Only) 7

Step 5: Clean Up Cookies on Rejection (Recommended) 8

Step 6: Verify Your Setup 9

Privacy Consent API Quick Reference 10

Common Mistakes 10

Support 11

Overview

The Seqrite Cookie Consent Manager lets you display a GDPR/DPDP-compliant consent banner on your website and actively enforce visitor choices. It generates a ready-to-embed package of files for your site.

What will you receive?

After configuring and publishing your consent manager, you can download a ZIP file containing four files:

- **seqrite.js**: Seqrite Consent Engine (do not modify)
- **seqrite-config.js**: Your consent configuration + banner auto-display
- **seqrite-config.css**: Banner styling
- **seqrite-enforcement.js**: Enforcement API + Google Consent Mode v2 signals

How does enforcement work?

The banner records visitor choices. The enforcement script (seqrite-enforcement.js) then blocks or activates tracking scripts on your page based on those choices. Scripts blocked by the consent engine remain inert until the visitor grants consent for the matching category. Choices are stored for 365 days in a browser cookie.

Applicable Steps Based on Your Setup/ Follow the Steps That Apply to You

Read through the questions below and follow only the sections that apply to your website.

Question 1 — Do you use Google Tag Manager (GTM)?

- YES → Follow Steps 1, 2, 3 (GTM Blocking), 4 (GTM Triggers), and 5.
- NO → Follow Steps 1, 2, and 5.

Question 2 — Are your tracking scripts written directly in HTML?

- YES → Use Approach A (markup-based) in Step 2.
- NO / Single-Page App (React, Angular, Vue) → Use Approach B (API-based) in Step 2.

Question 3 — Do you embed YouTube, Vimeo, or other iframes?

- YES → Follow Step 2C (iframe blocking).

Step 1: Embed the Four Scripts

Upload all four files from the downloaded ZIP to your web server or CDN. Then add the following code inside the <head> section of every page on your website, in this exact order. Load order is critical — seqrite-enforcement.js MUST be last.

```
<!-- Step 1a: Consent banner styles -->
<link rel="stylesheet" href="/path/to/seqrite-config.css"/>

<!-- Step 1b: Seqrite Consent Engine (must load before config) -->
<script defer src="/path/to/seqrite.js"></script>

<!-- Step 1c: Consent configuration + auto-show banner -->
<script defer src="/path/to/seqrite-config.js"></script>

<!-- Step 1d: Enforcement API (MUST be last) -->
<script defer src="/path/to/seqrite-enforcement.js"></script>
```

Important:

- Replace /path/to/ with the actual URL path where you uploaded the files on your server or CDN.
- Do NOT change the order. If seqrite-enforcement.js loads before seqrite-config.js, enforcement will not activate.

Step 2: Gate Your Tracking Scripts

Once the four scripts are embedded, you must mark which of your tracking scripts should be consent-gated. There are two approaches. Choose the one that matches your setup or combine both.

- **Approach A — Markup-Based (Static HTML)**
  Use this approach if your tracking scripts appear as <script> or <iframe> tags written directly in your HTML. The consent engine intercepts these elements at page load and activates them only when the visitor has consented to the matching category. For each tracking script, make two changes:

  1. Change type = **"text/javascript"** to type=**"text/plain"** so the browser does not execute it.
  2. Add data-type and data-name attributes so the consent engine knows which consent category controls it.

  Note: The data-name value must match the category name in your seqrite-config.js (for example, analytics, marketing, functional). You can check the exact names by opening **seqrite-config.js** and looking at the services array.

Google Analytics / GA4

```
<!-- BEFORE: loads unconditionally -->
<script async src="https://www.googletagmanager.com/gtag/js?id=G-
XXXXXXXXXX"></script>
<script>
  window.dataLayer = window.dataLayer || [];
  function gtag(){dataLayer.push(arguments);}
  gtag("js", new Date());
  gtag("config", "G-XXXXXXXXXX");
</script>

<!-- AFTER: blocked until visitor consents to "analytics" -->
<script type="text/plain"
        data-type="text/javascript"
        data-name="analytics"
        data-src="https://www.googletagmanager.com/gtag/js?id=G-XXXXXXXXXX">
</script>
```

```
<script type="text/plain"
        data-type="text/javascript"
        data-name="analytics">
  window.dataLayer = window.dataLayer || [];
  function gtag(){dataLayer.push(arguments);}
  gtag("js", new Date());
  gtag("config", "G-XXXXXXXXXX");
</script>
```

Facebook / Meta Pixel

```
<!-- AFTER: blocked until visitor consents to "marketing" -->
<script type="text/plain"
        data-type="text/javascript"
        data-name="marketing">
  !function(f,b,e,v,n,t,s)

{...}(window,document,"script","https://connect.facebook.net/en_US/fbevents.js")
;
  fbq("init", "YOUR_PIXEL_ID");
  fbq("track", "PageView");
</script>
```

HotJar

```
<!-- AFTER: blocked until visitor consents to "analytics" -->
<script type="text/plain"
        data-type="text/javascript"
        data-name="analytics">
  (function(h,o,t,j,a,r){
    h.hj=h.hj||function(){(h.hj.q=h.hj.q||[]).push(arguments)};
    h._hjSettings={hjid:YOUR_HOTJAR_ID,hjsv:6};
    a=o.getElementsByTagName("head")[0];
    r=o.createElement("script");r.async=1;
    r.src=t+h._hjSettings.hjid+j+h._hjSettings.hjsv;
    a.appendChild(r);
```

```
  })(window,document,"https://static.hotjar.com/c/hotjar-",".js?sv=");
</script>
```

Generic inline script

Any custom inline tracking code follows the same pattern:

```
<script type="text/plain"
        data-type="text/javascript"
        data-name="analytics">
  /* your tracking code here */
</script>
```

- **Approach B — API-Based (Single-Page Apps / Dynamic Scripts)**
  Use this approach if your scripts are loaded dynamically at runtime (React, Angular, Vue, or any JavaScript that calls fetch/import/createElement). Use PrivacyConsent.watch() to be notified whenever the visitor accepts or rejects a category, then load or stop your scripts accordingly.

  Add the following code after your seqrite-enforcement.js script tag:

```
<script>
document.addEventListener("DOMContentLoaded", function() {
  if (!window.PrivacyConsent) return;

  // Check consent for visitors who already consented (returning visitors)
  if (PrivacyConsent.getConsent("analytics")) {
    loadGoogleAnalytics();  // replace with your function
  }
  if (PrivacyConsent.getConsent("marketing")) {
    loadFacebookPixel();    // replace with your function
  }

  // React to new consent decisions (first-time visitors or changes)
  PrivacyConsent.watch(function(consents) {
    if (consents["analytics"] === true)  { loadGoogleAnalytics(); }
    if (consents["marketing"] === true)  { loadFacebookPixel(); }
    if (consents["functional"] === true) { loadChat(); }
  });
});
</script>
```

Note: The service names in getConsent() must match the category names in your seqrite-config.js (For example, analytics, marketing, functional). These are the same names used in data-name for Approach A.

Approach C: Blocking iframes (YouTube, Vimeo, Maps, etc.)
Embedded iframes (YouTube, Vimeo, Google Maps) load external content and can set third-party cookies. The consent engine can block these until the visitor consents. Change src to data-src and add a data-name attribute:

```
<!-- BEFORE: YouTube embed loads immediately -->
<iframe src="https://www.youtube.com/embed/VIDEO_ID"
        width="560" height="315" frameborder="0" allowfullscreen>
</iframe>

<!-- AFTER: blocked until visitor consents to "functional" -->
<iframe data-src="https://www.youtube.com/embed/VIDEO_ID"
        data-name="functional"
        width="560" height="315" frameborder="0" allowfullscreen>
</iframe>
```

Apply the same pattern for Vimeo, Google Maps, and other iframes:

```
<!-- Vimeo -->
<iframe data-src="https://player.vimeo.com/video/VIDEO_ID"
        data-name="functional">
</iframe>

<!-- Google Maps -->
<iframe data-src="https://www.google.com/maps/embed?pb=..."
        data-name="functional">
</iframe>
```

Note: A contextual overlay is shown where the iframe would appear, informing visitors that they must consent to view the embedded content. The overlay disappears automatically once consent is granted.

Step 3: Block the GTM Container (GTM Sites Only)

Google Tag Manager loads a single container script that then injects all other scripts (GA4, Facebook Pixel, Hotjar, etc.) at runtime. Because those injected scripts are not present as static HTML tags, the consent engine cannot intercept them directly. You must block the GTM container itself so that no tags fire before consent.

Note: Remove the standard GTM snippet entirely, do NOT keep both the old snippet and the new blocked version. Having both will cause double-firing of all tags.

Replace the standard GTM head snippet with this consent-gated version. Place it AFTER the four seqrite scripts:

```
<!-- REMOVE the standard GTM snippet: -->
<!-- (function(w,d,s,l,i){w[l]=w[l]||[];w[l].push({...})
   ...})(window,document,"script","dataLayer","GTM-XXXX"); -->

<!-- REPLACE with this (paste after Step 1 scripts in <head>): -->
<script type="text/plain"
        data-type="text/javascript"
        data-name="analytics"
        data-src="https://www.googletagmanager.com/gtm.js?id=GTM-XXXX">
</script>

<!-- Also replace the <body> noscript fallback: -->
<noscript>
  <iframe data-src="https://www.googletagmanager.com/ns.html?id=GTM-XXXX"
          data-name="analytics"
          height="0" width="0" style="display:none;visibility:hidden">
  </iframe>
</noscript>
```

Note: Replace GTM-XXXX with your actual GTM Container ID. If GTM was scanned into a category other than **analytics** in your consent manager, use that category name instead (check seqrite-config.js services array).

Step 4: Configure GTM Custom Event Triggers (GTM Sites Only)

After blocking the GTM container, you must configure GTM to activate your tags only when the visitor consents. seqrite-enforcement.js automatically pushes consent events to window.dataLayer every time a visitor accepts or rejects a category. You need to create Custom Event Triggers in your GTM workspace that listens for these events.

Events pushed automatically by seqrite-enforcement.js:

```
privacyconsent-analytics-accepted
privacyconsent-analytics-rejected
privacyconsent-marketing-accepted
privacyconsent-marketing-rejected
privacyconsent-functional-accepted
privacyconsent-functional-rejected
```

4.1 Create a Trigger for Each Category

1. In your GTM workspace, go to **Tags > Triggers > New**.
2. Choose Trigger Type: Custom Event.
3. Set Event Name to: privacyconsent-analytics-accepted
4. Set **This trigger fires on**: All Custom Events.
5. Name the trigger **Seqrite: Analytics Accepted** and click **Save**.
6. Repeat for each category you use:
   - privacyconsent-marketing-accepted → name: **Seqrite: Marketing Accepted**
   - privacyconsent-functional-accepted → name: **Seqrite: Functional Accepted**

4.2 Assign Triggers to Your Tags

1. Open your GA4 Configuration tag.
2. Under Triggering, remove the existing Page View trigger.
3. Add trigger: **Seqrite: Analytics Accepted** and **Save**.
4. Open your Facebook Pixel tag and assign trigger: **Seqrite: Marketing Accepted**.
5. Open any other analytics tags (Hotjar, etc.) and assign the relevant trigger.
6. Publish your GTM container.

4.3 Test in GTM Preview Mode

1. Open **GTM workspace > Preview**. Enter your website URL and click **Connect**.
2. On your website, do NOT consent yet. Verify that GA4 tag does NOT appear in the GTM debug panel.
3. Accept Analytics consent. Verify that privacyconsent-analytics-accepted event appears in the GTM debug panel and that the GA4 tag fires.
4. Reject Analytics. Verify that privacyconsent-analytics-rejected event appears and GA4 does NOT fire again.
5. Publish your GTM workspace changes once testing is complete.

Step 5: Clean Up Cookies on Rejection (Recommended)

When a visitor withdraws consent, seqrite-enforcement.js stops any future script activation. However, cookies that were already set by previously consented scripts are not automatically deleted — the browser does not allow a script to forcibly delete cookies set by external domains. You should add cleanup logic to delete first-party copies of those cookies.

Add the following code after your seqrite-enforcement.js script tag:

```
<script>
document.addEventListener("DOMContentLoaded", function() {
  if (!window.PrivacyConsent) return;

  PrivacyConsent.watch(function(consents) {

    // Clean up Google Analytics cookies when analytics is rejected
    if (consents["analytics"] === false) {
      ["_ga", "_gid", "_gat", "_ga_XXXXXXXXXX"].forEach(function(name) {
        document.cookie = name + "=; expires=Thu, 01 Jan 1970 00:00:00 UTC;
path=/;";
        document.cookie = name + "=; expires=Thu, 01 Jan 1970 00:00:00 UTC;
path=/;",
          + " domain=." + location.hostname + ";";
      });
    }

    // Clean up Facebook / Meta Pixel cookies when marketing is rejected
    if (consents["marketing"] === false) {
      ["_fbp", "_fbc"].forEach(function(name) {
        document.cookie = name + "=; expires=Thu, 01 Jan 1970 00:00:00 UTC;
```

```
path=/;";
        document.cookie = name + "=; expires=Thu, 01 Jan 1970 00:00:00 UTC;
path=/;",
          + " domain=." + location.hostname + ";";
      });
    }

    // Add cleanup blocks for any other cookies your site sets

  });
});
</script>
```

Note: Replace _ga_XXXXXXXXXX with your actual GA4 measurement-ID cookie name. You can find cookie names in browser **DevTools > Application > Cookies**.

Step 6: Verify Your Setup

After completing the steps above, use your browser DevTools to confirm that enforcement is working correctly. Clear all cookies before each test to start from a clean state.

6.1 Network Tab

- First visit with no consent: requests to google-analytics.com, connect.facebook.net, or static.hotjar.com should NOT appear.
- First visit (GTM blocked): gtm.js request should NOT appear before consent.
- After accepting Analytics: googletagmanager.com/gtag/js request appears within seconds.
- After accepting Analytics (GTM): gtm.js appears, then GA4 requests follow.

6.2 Application > Cookies

- Before any consent: _ga, _gid, _fbp, _hjid cookies must NOT exist.
- After accepting Analytics: _ga, _gid appear.
- After rejecting Analytics (with cleanup code): _ga, _gid are deleted.

6.3 Browser Console

Type these commands in the browser console to verify the enforcement API:

```
PrivacyConsent.getConsent("analytics")      // → false (before consent)
PrivacyConsent.services                     // → list of detected services
PrivacyConsent.getConsentByCategory("analytics")  // → true or false
window.dataLayer                            // → inspect for privacyconsent-*
events
```

6.4 GA4 DebugView (Consent Mode v2)

- Before consent: GA4 DebugView shows **Consent denied**, no user-identifying data is sent.
- After accepting Analytics: DebugView shows analytics_storage: granted.

Note: seqrite-enforcement.js automatically sets Google Consent Mode v2 default signals to **denied** on page load. When the visitor accepts analytics or marketing, the signals are updated to **granted** and sent to Google. This happens automatically; no additional code is needed.

Privacy Consent API Quick Reference

The enforcement script exposes window.PrivacyConsent with the following methods. All methods are safe to call from any page script after seqrite-enforcement.js has loaded.

Method	What it does
`PrivacyConsent.getConsent("serviceName")`	Returns true/false — was this service/category consented?
`PrivacyConsent.getConsentByCategory("cat")`	Returns true if ALL services in this category are consented
`PrivacyConsent.watch(callback)`	Register a function called on every consent change; receives a consents object
`PrivacyConsent.unwatch(callback)`	Unregister a previously registered watch callback
`PrivacyConsent.acceptAll()`	Programmatically accept all services and save
`PrivacyConsent.rejectAll()`	Programmatically reject all services and save
`PrivacyConsent.show()`	Re-show the consent banner
`PrivacyConsent.showModal()`	Open the full consent modal
`PrivacyConsent.updateConsent("name", true/false)`	Update consent for one service
`PrivacyConsent.saveAndApply()`	Save and apply all pending consent changes
`PrivacyConsent.resetAll()`	Clear all stored consent decisions
`PrivacyConsent.services`	Array of detected services with {name, category} for each
`PrivacyConsent.updateGoogleConsentMode(consents)`	Manually push Google Consent Mode v2 update

Common Mistakes

- **Scripts embedded out of order**
  seqrite-enforcement.js MUST be the last of the four scripts. If it loads before seqrite-config.js, the consent engine is not ready, and enforcement silently does nothing.
- **Data-name does not match the category name**
  The data-name value on your <script> tags must exactly match the service/category name in seqrite-config.js. Open the file and look for the services array to confirm the correct names

(For example, analytics, marketing, functional).

- **GTM — standard snippet not removed**
  If you added the consent-gated GTM snippet but did not remove the original GTM snippet, GTM loads twice. Remove the original snippet entirely.

- **Consent not cleared between tests**
  Consent decisions are stored in a browser cookie. If you accepted consent in a previous test session, the banner will not show again until the cookie expires. Clear all site cookies in DevTools > Application > Cookies before retesting.

- **Cleanup code runs before enforcement.js initializes**
  Place PrivacyConsent.watch() calls inside a DOMContentLoaded listener or ensure they run after seqrite-enforcement.js has loaded.

Support

If you need assistance with integration, contact the Seqrite support team and provide your Consent Manager ID (visible in the admin portal URL) along with a description of the issue. For technical debugging, include a screenshot of browser DevTools Network and Console tabs captured before and after consenting.