DP24 – Cross-Layer Resilience and Compromise Containment

When something beneath the Meta-Layer breaks, the damage stops at the boundary



1. Purpose of This Draft

This draft articulates Desirable Property 24 (DP24) as the Meta-Layer's requirement that identity, authority, interactions, and governance keep their integrity when the websites, browsers, devices, networks, services, overlays, or AI systems the Meta-Layer depends on are compromised.

The Meta-Layer runs on top of systems it does not control. DP24 defines how the Meta-Layer relates to each of those layers: what it may rely on, what it must verify, and what it must never inherit. It also defines how several Meta-Layer clients share the same page and screen without competing for it, and how annotations stay attached to what their authors meant when the page beneath them changes or cannot support them.

If DP1 defines who can act, DP13 defines how AI behavior is bounded, and DP15 defines how artifacts and software prove their integrity, DP24 defines how those guarantees survive when a neighboring layer turns hostile. It is complementary to those properties, not a replacement for them.

DP24 does not prescribe a browser, extension framework, operating system, or security vendor. It defines the trust boundaries any implementation has to declare and defend.

---

2. Problem Statement

The Meta-Layer is a shared coordination environment above today's web. It lets people, communities, and AI agents meet and work together across platforms that were never designed to cooperate.

The web beneath it was not designed to provide the identity, accountability, isolation, and agent containment such an environment needs.

2.1 Inherited trust

Websites are compromised. Browsers ship vulnerabilities. Software supply chains are poisoned. AI agents meet malicious instructions disguised as ordinary content, a pattern now catalogued as the leading risk for applications built on large language models (OWASP Top 10 for LLM Applications, LLM01: Prompt Injection).

A layer that simply inherits the authority and weaknesses of these systems becomes one more attack surface rather than a trustworthy superstructure.

2.2 Interoperability as an amplifier

The Meta-Layer connects participants, agents, information, and communities across platform boundaries. A successful attack on one site can therefore travel further than that site.

The same interoperability that makes collective intelligence possible can also amplify attacks unless clear security boundaries are established.

2.3 A shared screen without shared rules

Meta-Layer clients appear on top of pages they do not own: browser extensions, web embeds, and SDK integrations. Each one typically adds its own floating launch icon or panel.

Nothing today coordinates them. When a participant runs two or three overlays, their launch icons stack in the same corner, cover one another, cover the host page's own controls, and fight over which one is on top. A host page or a hostile extension can also draw an icon that looks exactly like a trusted one (OWASP: Clickjacking).

Recurring failure patterns include:

- **inherited authority**, where content gains power simply because the Meta-Layer displayed or relayed it
- **boundary collapse**, where one compromised component reaches everything connected to it
- **silent continuation**, where a system keeps operating after a security assumption has stopped holding
- **surface contention**, where overlays occlude, imitate, or crowd one another and the host page
- **anchor instability**, where the page beneath an annotation changes, hides, or rewrites the passage the annotation refers to

DP24 reframes security as a property of the relationships between layers, not of any single component. The question is not whether a lower layer will fail, but what that failure is allowed to reach.

---

3. Threats and Failure Modes

DP24 assumes every layer beneath and beside the Meta-Layer will at some point fail or become adversarial.

3.1 Host page impersonation and interface redressing

A compromised or malicious website draws convincing copies of Meta-Layer interface elements, or places invisible elements over real ones, to capture clicks, credentials, or consent.

Example: A news page renders a fake sign-in prompt styled exactly like the participant's overlay and collects their recovery phrase.

Why this matters: If participants cannot tell the Meta-Layer's surfaces from the page's, every protection behind those surfaces can be bypassed by asking the participant directly.

3.2 Content-borne instruction injection

Text, images, or metadata on a page carry instructions aimed at AI agents operating in the Meta-Layer.

Example: A product review contains hidden text telling any summarizing agent to post the participant's private notes to a public thread.

Why this matters: An agent that treats encountered content as instruction lets any page author borrow the agent's authority.

3.3 Credential and session extraction

A lower-trust execution context – the host page, a shared script, a compromised embed – reads tokens, keys, or session state that belong to the Meta-Layer.

Example: An overlay stores a session token where the host page's scripts can read it. A compromised advertising script on that page copies it and replays it from elsewhere.

Why this matters: Stolen credentials turn a single compromised page into persistent control over a participant's identity across every community they belong to.

3.4 Sibling overlay interference

Another extension or overlay running on the same page – malicious, compromised, or simply careless – covers, imitates, or reads the interface of a Meta-Layer client.

Example: A coupon extension places its launch icon exactly over a community overlay's icon and opens its own panel when the participant meant to open their community conversation.

Why this matters: Compromise does not come only from below. Peers at the same layer share the same page and can interfere with one another unless coexistence has rules.

3.5 Client supply-chain and update compromise

The Meta-Layer client itself – an extension, SDK, or embed script – is replaced by a malicious version through a hijacked publisher account, a poisoned dependency, or an automatic update.

Example: An overlay extension's publisher account is phished, and a new version that forwards every message the participant reads is pushed to all installations overnight.

Why this matters: A trusted client that turns hostile carries all the authority participants granted it. DP15 defines supply-chain transparency; DP24 defines how far a compromised client can reach.

3.6 Federation propagation

A compromised federated service or community uses its legitimate connections to influence, impersonate, or extract from other communities.

Example: A community server whose admin keys were stolen sends forged moderation decisions to every community that federates with it.

Why this matters: Federation multiplies reach. Without boundaries, one community's compromise becomes everyone's.

3.7 Silent continuation under broken assumptions

A system keeps operating normally after a dependency it relies on for security – an attestation service, a key server, a revocation list – becomes unreachable or untrustworthy. This failure is usually not malicious.

Example: A revocation service goes offline, and clients keep accepting a revoked agent credential because the check fails open.

Why this matters: The most damaging failures are often the ones nobody notices. A system that cannot tell a safe state from an unsafe one should not act as though it can.

3.8 Launcher crowding and accidental occlusion

Several well-behaved overlays each place a launch icon in the same screen region, covering one another and the host page's own controls. This failure is not malicious.

Example: A participant runs three Meta-Layer clients. All three place their icon in the bottom-right corner, over each other and over the site's chat button, and the participant can reach only the top one.

Why this matters: Contention teaches participants to click whatever is on top. That habit is exactly what impersonation attacks exploit.

3.9 Downward harm

The Meta-Layer becomes the attacker: an overlay injects scripts that break the host page, leaks data between sites, or carries an attack from one page to another.

Example: An overlay copies the page's form contents into a shared context visible on other sites the participant visits.

Why this matters: The relationship between layers runs in both directions. A Meta-Layer that harms the sites beneath it loses the legitimacy it needs to operate on them.

3.10 Anchor spoofing

A host page controls the text that Meta-Layer annotations attach to. It can show one version of a passage to the participant who writes a note and another version to everyone who reads it later, or rewrite the text around an existing note so that the note appears to say something its author never meant.

Example: A participant anchors "this claim is false" to a sentence on a product page. The site later replaces the sentence with a competitor's claim, keeping the same wording at the start and end so the anchor still matches. Readers now see the participant appearing to call the competitor a liar.

Why this matters: An anchor is a statement about someone else's page. If the page can change what the anchor points at without anyone noticing, the page can put words in a participant's mouth.

3.11 Silent anchor drift

Pages change for ordinary reasons: edits, redesigns, personalization, rotating content, re-rendering by frameworks. When an anchor no longer matches exactly, a client may reattach the note to the closest match it can find. This failure is usually not malicious.

Example: A news article is updated with a correction. A note on the original paragraph reattaches to a nearby paragraph that says something different, and nothing tells readers it moved.

Why this matters: A note attached to the wrong passage is worse than a missing note. It misleads readers and misrepresents its author, and it does so without any visible sign of failure.

---

4. Core Principle

The Meta-Layer preserves the integrity of identity, authority, interactions, and governance even when the systems it depends on are compromised, within explicitly declared trust boundaries and threat assumptions.

> Compromise of a lower layer must not automatically confer control over the Meta-Layer.

Example: A participant visits a compromised news site while taking part in a signed-in Meta-Layer conversation about the page. The site tries to impersonate other participants, extract credentials, alter the overlay, and steer an AI agent. It remains an untrusted source of content. Its scripts gain no authority over the participant's identity, private conversations, community resources, or agent permissions. Where one attempt does succeed against a component, the damage is contained, evidence is preserved, and recovery begins.

What this feels like: Participants can visit any page, including a hostile one, without worrying that the page can speak for them, read their communities, or redirect their agents.

Without this: Every website becomes a potential administrator of the Meta-Layer, and the safest choice is not to use it on any page that matters.

---

5. Primary Mechanisms and Structural Conditions

Cross-layer resilience comes from boundaries that hold regardless of what arrives through them. No single control is sufficient. Each mechanism below assumes the others can fail.

The layers and their relationships are defined in Section 6. Rules for sharing the page and the screen are in Section 7. Page compatibility and anchor integrity are in Section 8.

5.1 Authority independent of content origin

Authority derives from authenticated participants (DP1) and community governance (DP12), never from content. Data arriving through websites, embedded applications, external services, and AI outputs is information, not instruction.

- Meta-Layer services MUST authorize every privileged action on the server side, independently of webpage content and of claims supplied by the client.
- Meta-Layer clients MUST NOT grant privileges to content because it was displayed, interpreted, or relayed.
- Agent runtimes MUST treat page content and tool outputs as untrusted data, separate from the instructions of the agent's accountable principal.

5.2 Isolation of credentials and privileged operations

Credentials, keys, and privileged operations live in execution contexts that lower-trust code cannot reach.

- Meta-Layer clients MUST keep credentials and session material out of any context the host page’s scripts can read.
- Meta-Layer clients SHOULD bind sessions to the device or client instance, so a copied token cannot be replayed elsewhere.
- Meta-Layer services MUST be able to revoke a session, key, or delegation within minutes, and clients MUST honor revocation.

5.3 Bounded agent authority

AI agents act through enforceable, scoped, and revocable delegation tied to an accountable person or organization. DP13 defines the containment dimensions; DP24 adds that no lower layer can widen them.

- Meta-Layer services MUST check every agent action against the agent’s authenticated delegation at the point of execution.
- A malicious page, compromised service, or manipulated model MUST NOT be able to expand an agent’s authority beyond its legitimate delegation.

5.4 Integrity of identity, records, and governance

Compromised lower layers cannot silently forge identities, alter protected records, falsify authorization, or change governance outcomes.

- Meta-Layer services MUST sign or otherwise integrity-protect community records and governance decisions so that tampering is detectable (DP15).
- Where compromise cannot be prevented, systems SHOULD let participants and communities challenge a questionable action and reconstruct an accountable record of it (DP22).

5.5 Containment and recovery

The reach of any compromise is bounded, identifiable, and proportionate to the authority the compromised component actually held.

- Federated services MUST NOT accept privileges asserted by another community without verifying them against that community’s published keys and policies.
- Implementations MUST define containment and recovery procedures for detected compromise: isolate, revoke, notify affected participants, restore.
- Security-critical operations MUST produce audit evidence protected against tampering and disclosed in proportion to privacy (DP4).

5.6 Graceful degradation

When a dependency becomes unavailable or untrustworthy, the Meta-Layer keeps as much safe function as possible and says what it has lost.

- Meta-Layer clients MUST fail closed for privileged actions when a security check cannot be completed.
- Meta-Layer clients SHOULD keep low-risk functions, such as reading public content, available while privileged ones are suspended.
- Systems MUST show participants when protections are reduced, and which ones.

5.7 Verifiable security claims

Resilience is demonstrated, not asserted.

- Implementations MUST publish their threat model: the threats considered, the boundaries tested, the results observed, and the limitations that remain.
- Implementations SHOULD submit to adversarial testing and independent review, with reproducible evidence.

5.8 Federated protection sharing

Communities share what they learn without surrendering autonomy to a central authority.

- Communities SHOULD be able to publish and consume verified vulnerability advisories and remediation guidance through a common, signed format.
- Each community MAY decide which advisories to act on, and SHOULD record that decision.

5.9 Anchor integrity and declared page capability

An annotation stays attached to what its author meant, or says plainly that it no longer is. Section 8 describes how pages break anchors and what participants see.

- Meta-Layer clients MUST store enough about each anchor – the quoted text with its surrounding context, and a fingerprint of the passage when the note was made – to detect that the passage has changed.
- Meta-Layer clients MUST NOT silently reattach an annotation to different text. When the match falls below a declared confidence, the annotation is shown as detached, with its original quote.
- Meta-Layer clients MUST show readers when the passage under an annotation has changed since it was written.

- Meta-Layer clients MUST tell participants, before they write, when a page supports only page-level notes or none at all.
- Meta-Layer clients SHOULD suppress in-page overlays, or keep them read-only, on sign-in, payment, and similar sensitive pages.

---

6. Layer Model and Trust Relationships

This section is a reference model. It names the layers the Meta-Layer runs on and beside, and states what each relationship may and may not carry. Implementations declare their own boundaries against it.

6.1 The layers

Layer	Examples	What the Meta-Layer relies on it for
Device and operating system	Laptop, phone, OS	Local execution, storage, input, and display
Network and infrastructure	ISP, DNS, CDN, hosting	Transport and availability
Browser	Chrome, Firefox, Safari	Process isolation, extension sandboxing, origin separation
Host website and content	News sites, apps, embedded media	The page being discussed – as content only
Meta-Layer clients	Extensions, web embeds, SDK integrations	Rendering the Meta-Layer for a participant on a page
Meta-Layer services	Identity, communities, messages, governance records	Authoritative state and authorization
Federation	Other communities and their services	Shared identity, content, and decisions across communities

AI agents and third-party services are not a layer of their own. They act within one of these layers, with authority delegated by an accountable principal (Section 5.3).

6.2 Authority flows from principals, never from content

Authority originates with authenticated participants and community governance. It flows down to the clients and agents acting on their behalf.

It never flows up from lower layers. A host page, a network path, or a content item can supply information to the Meta-Layer, but cannot grant itself or anyone else the right to act in it.

Example: A host page sends a message to an overlay claiming the participant has approved a post. The overlay treats it as a request and asks the participant, because only the participant's own action, verified by Meta-Layer services, can approve it.

6.3 Relationships between layers

Each relationship between adjacent layers has a defined limit.

Relationship	The upper layer may rely on	The upper layer must never inherit
Client ← device and OS	Execution and display, which cannot be avoided	Assurance that a compromised device is safe (declared out of scope)
Client ← browser	Sandboxing and origin isolation	Assumptions beyond the browser's documented guarantees
Client ← host website	The page's content, as data	Any authority, credential access, or ability to speak for the participant
Services ← client	Authenticated requests	Trust in client-supplied claims about permissions or identity
Community ← federated community	Signed assertions that can be verified	Privileges the other community cannot prove under its own policies
Agent ← content	Information to summarize or analyze	Instructions or expanded authority

6.4 Relationships between peers

Some of the riskiest relationships are between peers at the same layer: several Meta-Layer clients on one page, or several communities in one federation.

- Peers coordinate through declared interfaces only. No client reads another client's interface, storage, or messages.
- Peers do not imitate one another. A client that copies another's icon, name, or visual identity is non-conforming.
- Coordination conveniences between peers, such as the shared launcher in Section 7, never become trust anchors.

6.5 Obligations downward

The Meta-Layer has duties to the layers beneath it, not only protections against them.

- Meta-Layer clients do not break, alter, or slow the host page beyond what rendering the overlay requires.
- Meta-Layer clients do not carry data from one site into another site’s context without the participant’s explicit action.
- Meta-Layer clients respect regions the host page declares as reserved for its own controls (Section 7.3).

6.6 The protection boundary

Some compromises cannot be defended against at the Meta-Layer. A compromised operating system controls local execution. A disrupted network prevents communication. A host page can always draw lookalike elements inside its own area.

Each implementation states which layers it treats as trusted, which it defends against, and which are outside its protection boundary. A participant can read that statement before relying on the implementation.

---

7. Shared Overlay Surfaces and Launcher Coexistence

Meta-Layer clients share two scarce things: the participant’s screen and the participant’s trust in what appears on it. This section sets rules so that several clients can coexist on one page without covering, crowding, or imitating one another.

7.1 One shared launcher per page

Conforming clients do not each inject their own floating launch icon. They register a launch entry with a single shared launcher on the page – a dock that lists every active Meta-Layer client.

- The first conforming client on a page hosts the launcher. Clients loaded later find it and register with it instead of drawing their own.
- Each registration declares the client’s name, icon, publisher, and requested actions.
- If the hosting client unloads, another registered client takes over hosting without losing the participant’s arrangement.

Example (hypothetical): A participant runs a community overlay, a translation overlay, and a note-taking overlay. Instead of three icons stacked in one corner, a single dock shows three entries. The participant opens any one with one click.

7.2 The participant arranges the launcher

The launcher belongs to the participant, not to any client.

- The participant chooses the launcher's edge and position, the order of entries, and which clients are shown, hidden, or pinned.
- The arrangement persists per participant and, where the participant chooses, per site.
- No client can move itself ahead of others, enlarge its own entry, or hide another client's entry.

7.3 Placement and collision rules

Where clients must draw outside the launcher – panels, highlights, inline tags – they follow shared placement rules.

- Each client declares the screen regions it intends to occupy. Clients do not overlap another client's declared region.
- Host pages can declare reserved regions, such as their own chat, consent, or checkout controls. Clients keep out of them.
- When regions conflict and cannot be negotiated, clients fall back to a deterministic order: the participant's arrangement first, then registration order. Raising one's own stacking order to win a conflict is non-conforming.
- When no conforming launcher is present, a client places its icon away from occupied regions and offers to join a launcher as soon as one appears.

7.4 Coordination is not trust

The page can see and imitate anything drawn inside it, including the shared launcher. The launcher coordinates placement. It does not prove identity.

- Actions that carry authority – signing in, approving, posting as the participant, granting agent permissions – happen in surfaces the host page cannot draw over or read, such as the browser's own side panel or toolbar.
- Clients give participants a way to confirm that a surface is genuine, such as a check reachable only from the browser's own interface.
- Launcher registrations are signed by the client's publisher, so a lookalike entry can be detected and flagged.

7.5 Accessibility and focus

Several overlays on one page also compete for keyboard focus and screen reader attention.

- The shared launcher is reachable by keyboard and announced once to assistive technologies, not once per client.
- Only one client panel holds focus at a time, and the participant can always return focus to the host page.
- Overlays follow DP21 and DP23 for modality and language access.

---

8. Page Compatibility and Anchor Integrity

Meta-Layer annotations attach to passages on pages the Meta-Layer does not control. Some pages cannot hold stable anchors, some block overlays entirely, and some should not carry overlays at all. This section names those cases and describes what participants see in each. The requirements are in Section 5.9.

8.1 How pages break anchors

Anchors fail for structural reasons, most of them ordinary:

- **re-rendered pages:** application frameworks replace page elements on every update, and generated class names change between releases, so anchors based on page structure stop matching
- **partially loaded pages:** long lists and feeds keep only the visible rows on the page, and collapsed or lazily loaded sections are absent until opened, so the anchored passage may not exist when the client looks for it
- **changing text:** relative timestamps, counters, rotating content, experiments, and personalized or regional versions mean two participants at the same address may not see the same words
- **edits and repeats:** the passage is revised, or the same words appear more than once on the page, so a quote alone cannot say which occurrence was meant
- **navigation without reloading:** single-page applications change the address without loading a new page, and a client that does not notice keeps showing the previous page’s annotations
- **reflow:** resizing, zoom, reader modes, and translation or theming extensions rewrite or move text, breaking any anchor based on screen position

- **no text to anchor:** text drawn on a canvas, content inside closed components or cross-origin frames, and media such as PDFs, images, and video need region or time anchors, not text anchors

The same content can also live at many addresses, through tracking parameters and mirrors, and different content can live at one address. A page's declared canonical address helps, but it is a claim made by the page.

8.2 Page capability levels

Each client classifies the current page and shows the result before the participant writes.

Level	What participants can do	Typical cause
Full	Anchor notes to passages	Stable text the client can match and fingerprint
Page-level only	Attach notes to the page as a whole	Text that is drawn, hidden, or changes on every visit
Read-only	Read existing notes; not write new ones	Sensitive pages (Section 8.4)
Unsupported	Nothing on the page; the side panel stays available	The browser or the page blocks the client, with the reason shown

Example: A participant opens a design tool that draws its text on a canvas. The overlay shows "Notes on this page attach to the page, not a passage" before the participant starts typing, instead of accepting a passage anchor that will fail for every reader.

Pages block clients for reasons outside the Meta-Layer: content security policies that refuse injected scripts (MDN: Content Security Policy), browser pages where extensions cannot run, enterprise policies that block extensions on certain sites, and scripts that remove anything they did not draw. Clients report these as the cause rather than failing silently.

8.3 Anchors that can be checked

An anchor records more than one way of finding its passage – the quoted text with words before and after it, its position, its place in the page's structure – following the selector model of the W3C Web Annotation Data Model (W3C, 2017). It also records a fingerprint of the passage at the time of writing.

- When every selector agrees, the annotation is shown normally.
- When the passage has changed since the note was written, readers see that it changed, and can see the original quote.

- When no selector finds the passage with enough confidence, the annotation is shown as detached, with its original quote, rather than moved to the nearest match.

This answers both anchor threats. Drift (Section 3.11) becomes visible instead of silent. Spoofing (Section 3.10) is detectable because the page can rewrite the passage, but cannot make the new version match the stored fingerprint.

8.4 Pages where overlays step back

On sign-in, payment, banking, health, and similar pages, in-page overlays are suppressed or read-only by default. These pages are where interface redressing (Section 3.1) and credential extraction (Section 3.3) do the most damage, and where an overlay is least needed.

Participants can still open the side panel, which the page cannot draw over.

Anchors on pages behind a sign-in or paywall quote content that other readers may not have access to. Clients show only as much of the quote as the reader could see for themselves, under DP4.

No web standard currently lets a site declare that it does not want to be annotated. Whether the Meta-Layer honors such a signal, and which one, is an open question (Section 19).

8.5 Learning which pages break

Re-anchoring outcomes are a signal. When clients report, in aggregate and without identifying participants, how often anchors on a site detach, communities can see which pages are unstable without waiting for complaints.

A shared, signed registry of page behaviors – sites that block overlays, sites whose text is drawn, sites that re-render constantly – lets every conforming client set the right capability level on the first visit. The registry follows the same signing and contestability rules as security advisories (Section 5.8).

---

9. Tensions and Tradeoffs

9.1 Unavoidable dependencies

The objective is not absolute independence from the infrastructure beneath. Some dependencies cannot be removed.

The objective is to minimize inherited trust, prevent unauthorized elevation of lower-layer authority, contain compromise, preserve recoverability, and make residual risks explicit.

9.2 Security and openness

Strict boundaries make the Meta-Layer harder to extend. DP7 requires that third-party tools can plug in without prior permission. DP24 requires that they cannot inherit authority by doing so. Communities weigh security against accessibility, usability, decentralization, privacy, performance, and the ability of independent developers to innovate.

9.3 Shared launcher and client independence

A shared launcher reduces contention but asks independent clients to cooperate. It can also become a point of capture if one client controls how others appear. The launcher therefore belongs to the participant (Section 7.2), and its behavior is defined by an open interface rather than by any single client.

9.4 Evidence and privacy

Audit evidence makes compromise visible and recoverable. It also records what participants did. Evidence is kept in proportion to the risk it addresses, under DP4.

9.5 Annotation and the host site’s autonomy

Annotating a page without its owner’s permission is part of what makes the Meta-Layer independent. It also means the Meta-Layer depends on pages that owe it nothing and may change, block, or resist it. Showing a detached note honestly, or declining to anchor at all, is preferred to anchoring that only appears to work.

---

10. Governance, Accountability, and Agency Surfaces

Cross-layer protections matter only if participants and communities can see them and act on them.

Participants must be able to:

- see which Meta-Layer clients and agents are active on the current page, who publishes them, and what authority each holds
- confirm that a surface asking for a sign-in or approval is genuine
- arrange, hide, or remove any client’s launcher entry
- revoke a client’s, session’s, or agent’s authority immediately
- see when protections are reduced and which ones
- see what the current page supports – passage notes, page-level notes, reading only, or nothing – and why
- see when a note’s passage has changed or its anchor has detached, and read the original quote

- report a suspected compromise and receive a traceable response

Communities must be able to:

- quarantine a client version, federated service, or agent class pending review
- publish and consume signed security advisories
- review incidents with access to the relevant evidence
- decide which federated assertions they accept, and record why

Example: A participant sees an unfamiliar entry in the launcher claiming to be their community client. They open the browser's side panel, which shows that the entry's signature does not match the community client's publisher. They report it in one action, and the community quarantines the impersonating extension for its members.

---

11. Incentives and Power Analysis

Screen space and trust are both attention, and attention is valuable.

What must be visible:

- which client controls the shared launcher and how entries are ordered
- who issues security advisories and who acts on them
- who certifies, tests, or reviews implementations, and who pays them

Capture patterns include:

- **placement capture:** a client vendor pays or negotiates for a better position in the launcher or overrides the participant's arrangement
- **certification capture:** a single reviewer becomes the gatekeeper for what counts as secure
- **security as lock-in:** a platform uses resilience requirements to exclude independent clients
- **advisory weaponization:** a community issues false advisories to quarantine a competitor

DP24 expects ordering to remain under participant control, review to be plural and verifiable, and advisories to be signed, attributable, and contestable.

---

12. Community Signals Informing DP24

Recurring signals include:

- overlay launch icons from different extensions stacking on and covering one another and the host site’s own controls
- uncertainty about whether a sign-in prompt belongs to the overlay or to the page
- concern that AI agents can be steered by content on the pages they read
- distrust of security claims that cannot be checked
- worry that one compromised community or extension could reach all of a participant’s communities
- annotations failing to stay attached on pages without stable anchors

These signals are not usability complaints. They indicate that participants cannot yet tell which layer is speaking to them, or how far a failure in one layer can travel.

---

13. Evaluation Criteria

Each criterion is written so that a “no” identifies a specific defect. Each test documents its threat assumptions, attack path, observed outcome, mitigation, and remaining risks. A successful test is evidence for a particular protection, not for resistance to all possible attacks.

13.1 Boundary integrity

- Can a webpage forge a message or privileged command that Meta-Layer services accept?
- Can a webpage, embed, or sibling extension read authentication material from a Meta-Layer client?
- Can content on a page cause an AI agent to act beyond its delegated authority?

13.2 Containment

- Can a compromised federated service change records or decisions in another community?
- Can a protected community record be altered without authorization and without detection?
- After a simulated compromise, how long do revocation and participant notification take?

13.3 Degradation

- Does the system continue privileged operations when a critical security dependency is unavailable?
- Does it tell participants which protections are reduced?

13.4 Surface coexistence

- With three conforming clients loaded on one page, does any launch icon cover another or a host-declared reserved region?
- Can a sibling extension place an icon over a conforming client’s launcher entry, or imitate it, without being detected?
- Can every authority-bearing action be completed only from a surface the host page cannot draw over?

13.5 Anchor integrity

- When the anchored passage is edited, do readers see that it changed?
- When the passage is removed, is the annotation shown as detached rather than moved to other text?
- Can a test page that shows different text to the author and to later readers make a note appear attached to text its author never saw, without detection?
- On a page whose text is drawn on a canvas, or on a page that blocks the client, does the participant learn what the page supports before writing?

---

14. Implementation Patterns

14.1 Trusted surface separation

Authority-bearing interactions run in browser-owned surfaces, such as an extension side panel, while the in-page overlay shows only content and launch entries.

14.2 Server-side capability checks

Each privileged request carries a short-lived, scoped capability that the service verifies. The client’s own view of its permissions is never trusted.

14.3 Content tainting for agents

Agent runtimes label every input with its origin and keep untrusted content out of the instruction channel, satisfying Section 5.3.

14.4 Launcher registration protocol

A small open protocol for registering with, hosting, and handing over the shared launcher, with signed registrations (Section 7).

14.5 Signed advisory feeds

Communities publish advisories in a common signed format that others can subscribe to and act on independently (Section 5.8).

14.6 Multi-selector anchors with fingerprints

Each anchor stores a quote with its surrounding words, a position, and a structural path, following the W3C Web Annotation selector model, plus a fingerprint of the passage. The client re-anchors by trying each selector, scores its confidence, and shows the result (Section 8.3).

14.7 Page behavior registry

A shared, signed list of sites and the capability level each supports, fed by aggregate re-anchoring outcomes (Section 8.5).

---

15. Foresight and Failure Design

Resilience degrades over time unless it is designed to be maintained.

Likely failure paths:

- **assumption drift:** a declared threat model stops matching how the system is actually deployed
- **alert fatigue:** degradation notices appear so often that participants ignore them
- **launcher erosion:** clients gradually stop registering with the shared launcher and return to their own icons
- **advisory poisoning:** a compromised or captured source floods the federation with false advisories
- **selector rot:** a popular site redesigns, and thousands of its annotations detach at once
- **capability shift:** future AI agents or new browser capabilities change which layers can reach which

Design responses:

- review the threat model on a fixed schedule and on every major release
- make notices rare, specific, and actionable

- test launcher conformance in every client release
- keep the original quote of every annotation, so detached notes remain readable and can be reattached by people
- require advisories to be signed and rate-limited, and give communities the final decision
- treat each new class of agent or client capability as a new boundary to declare

---

16. Relationship to Other Desirable Properties

- **DP1 (Federated Authentication and Accountability)** establishes who can act and how actions are attributed; DP24 keeps that attribution intact when adjacent systems are compromised.
- **DP2 (Participant Agency)** gives participants real controls; DP24 makes sure the controls they see are genuinely the Meta-Layer's.
- **DP4 (Data Sovereignty and Privacy)** gives participants control of their information; DP24 keeps that control from being bypassed through lower layers.
- **DP7 (Simplicity and Interoperability)** lets tools plug in freely; DP24 ensures that interoperability does not transfer authority, and that plugged-in clients share the screen by rule.
- **DP11 (Safe and Ethical AI)** and **DP13 (AI Containment)** define acceptable agent behavior and its enforceable limits; DP24 makes sure those limits survive manipulation from lower-trust environments.
- **DP12 (Community-based AI Governance)** decides who sets the rules; DP24 keeps compromised layers from changing them.
- **DP14 (Trust and Transparency)** and **DP15 (Security and Provenance)** make information and software verifiable; DP24 adds isolation, containment, and recovery across the architecture.
- **DP21 (Multi-modal)** and **DP23 (Universal Participation)** set access requirements that shared overlays must meet.
- **DP22 (Civic Memory)** preserves accountable records; DP24 keeps them reconstructable after compromise, and keeps annotations attached to what their authors actually saw.

Without DP24, each of these properties holds only as long as every system beneath the Meta-Layer stays honest.

---

17. Non-Goals and Explicit Boundaries

DP24 does not:

- promise that the Meta-Layer stays secure or available when every layer beneath it is compromised
- protect participants on a device whose operating system is compromised
- prevent a host page from drawing lookalike elements inside its own area; it requires that authority-bearing actions happen where the page cannot
- mandate a browser, extension framework, attestation technology, or security vendor
- create a central certification authority
- guarantee that every page can be annotated; it requires that participants are told when one cannot
- replace DP13's containment dimensions or DP15's supply-chain and provenance requirements

DP24 defines the trust boundaries between the Meta-Layer and the systems it runs on and beside, and what must hold at each one.

---

18. Minimum DP24 Alignment

A DP24-aligned implementation meets all of the following:

- Meta-Layer services MUST authorize privileged actions independently of webpage content and client-supplied claims.
- Meta-Layer clients MUST keep credentials out of any context readable by host-page scripts or other extensions.
- Meta-Layer services MUST check agent actions against authenticated, scoped, and revocable delegation.
- Federated services MUST NOT accept privileges from another community without verifying them.
- Implementations MUST produce tamper-protected audit evidence for security-critical operations.
- Implementations MUST define and test containment and recovery procedures, including participant notification.
- Meta-Layer clients MUST fail closed for privileged actions when a security check cannot complete, and MUST tell participants.
- Meta-Layer clients MUST register with a shared launcher when one is present, and MUST NOT cover, reorder, or imitate another client's entry.

- Meta-Layer clients MUST perform authority-bearing actions only in surfaces the host page cannot draw over or read.
- Meta-Layer clients MUST detect when an anchored passage has changed, MUST NOT silently reattach an annotation to different text, and MUST show detached annotations with their original quote.
- Meta-Layer clients MUST tell participants what the current page supports before they write.
- Implementations MUST publish their threat model, protection boundary, limitations, and evidence of adversarial testing.

If any of these is missing, the implementation's security depends on the honesty of the systems beneath it.

18.1 Strong DP24 alignment

A strongly aligned implementation also binds sessions to devices, publishes reproducible adversarial test results, takes part in signed advisory sharing, contributes to a shared page behavior registry, and passes the surface coexistence tests in Sections 13.4 and 13.5 against other conforming clients.

18.2 Failure conditions

An implementation must not claim DP24 alignment if a webpage can trigger a privileged action, if credentials are readable from the page, if its client draws over or imitates another client's launcher entry, or if it moves annotations to different text without telling readers.

---

19. Open Questions and Future Work

These questions are not blockers. They are where reasonable implementers currently disagree.

- Which lower-layer compromises should the Meta-Layer be expected to withstand, and which are outside its protection boundary?
- What minimum guarantees should every conforming extension, SDK, web embed, and federated service provide?
- How can communities verify one another's security claims without a central certification authority?
- How can security improvements spread quickly across the federation while respecting local autonomy?
- What evidence is enough to claim that an implementation satisfies DP24?

- Can the shared launcher be standardized at the browser level, so the page cannot imitate it at all?
- How should the launcher behave on mobile and in-app browsers, where screen space is smallest?
- Should the Meta-Layer honor a signal from a site that does not want to be annotated, and if so, which signal, and with what exceptions?
- When participants at the same address see different content, whose version does an anchor describe?
- Who maintains the page behavior registry, and how is a site's listing contested?

---

20. Path Toward ML-RFC

- Publish a threat model template that implementations fill in against the layer model (Section 6).
- Specify the shared launcher registration and handover protocol, with signed registrations (Section 7).
- Build an adversarial test page and conformance suite covering the criteria in Section 13.
- Run the first demonstration: a deliberately malicious webpage attacking a signed-in Meta-Layer interaction, with two or more independent clients on the same page.
- Define a signed advisory format for federated vulnerability sharing.
- Specify the anchor format – selectors, fingerprint, confidence – and the page capability levels (Section 8), with a test corpus of pages known to break anchors.
- Iterate with extension, SDK, and federation implementers; progress is measured by working, tested systems.

Candidate requirements for the ML-REQ stage:

Theme	**Candidate requirement**
Content-independent authority	Privileged actions are authorized independently of untrusted webpage content and client-supplied claims
Credential isolation	Credentials and privileged operations are isolated from lower-trust execution contexts under the declared threat model
Agent authority	

Theme	Candidate requirement
	Agent actions are checked against authenticated, scoped, and revocable authority
Federation boundaries	Federation boundaries prevent unauthorized transfer of privileges between communities
Audit evidence	Security-critical operations produce protected audit evidence
Containment and recovery	Systems support defined containment and recovery procedures after detected compromise
Surface coexistence	Clients share a participant-controlled launcher and do not occlude or imitate one another
Anchor integrity	Annotations detect changes to their passage, are never silently reattached, and show their original quote when detached
Declared page capability	Clients tell participants what each page supports before they write
Published assumptions	Implementations publish their security assumptions, limitations, and adversarial testing evidence

---

21. Closing Orientation

DP24 decides whether the Meta-Layer is a trustworthy coordination layer or simply the web’s weaknesses reproduced one level higher.

When DP24 is strong, participants can take part on any page, including a hostile one, without the page speaking for them. Several Meta-Layer clients share the screen without covering one another, and a failure in one layer stops at its boundary.

When DP24 is weak, every website, extension, and federated peer is a potential administrator of the Meta-Layer.

Safe and accountable human and AI coexistence needs defensible boundaries between the Meta-Layer and the systems it runs on.

> The Meta-Layer must not simply rise above the web. It must demonstrably resist, contain, and recover from attacks originating beneath it.