

CAP after browser-side XSLT

What changes, what doesn't,
and how publishers can respond



Ian Ibbotson

CTO, Alert-Hub.org

ian.ibbotson@alert-hub.org

[Telegram: @CAP_community](https://t.me/CAP_community)

What Chromium is actually removing

CAP

stays

XML

stays

Browser XSLT

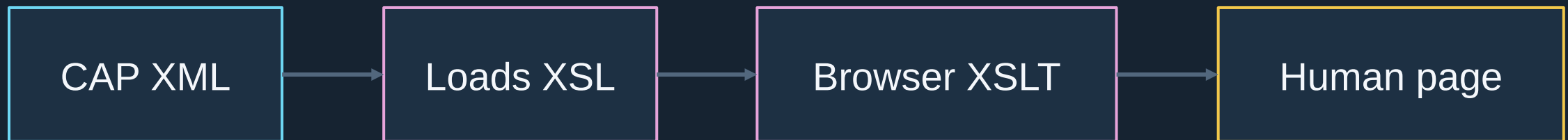
goes

Current plan: Chrome 158, 17 November 2026

Recheck before the event. Temporary exceptions exist.

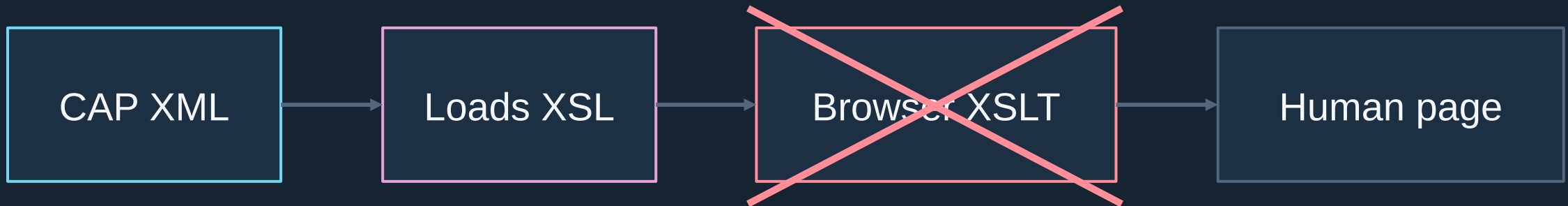
The familiar browser transformation

```
<?xml version="1.0"?>  
<?xml-stylesheet type="text/xsl" href="cap.xsl"?>  
<alert xmlns="urn:oasis:names:tc:emergency:cap:1.2">  
  ...  
</alert>
```



The server sends XML. The browser constructs the page.

One presentation mechanism disappears



CAP semantics unchanged

What disappears? A presentation mechanism.

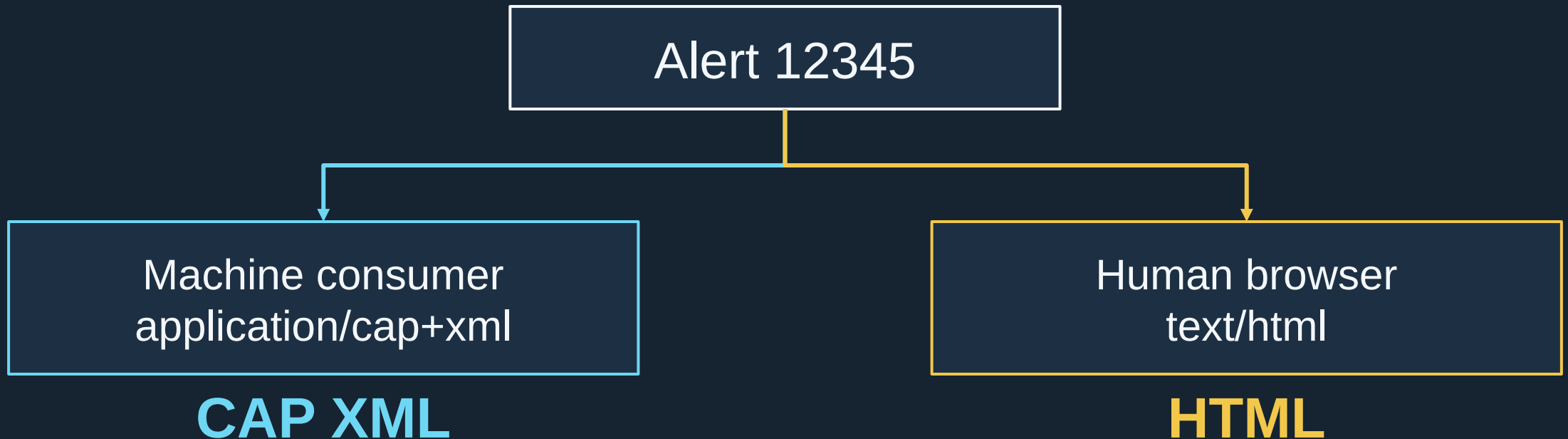
Where should presentation happen?



Choose the rendering location and the HTTP contract.

Resource $\neq$ representation

`https://alerts.example.org/alerts/12345`



The URL identifies the alert. The response represents it.

Three HTTP headers, two representations

MACHINE REQUEST

```
GET /alerts/12345
Accept: application/cap+xml
```

```
200 OK
Content-Type:
  application/cap+xml
Vary: Accept
```

```
<alert ...>
```

HUMAN REQUEST

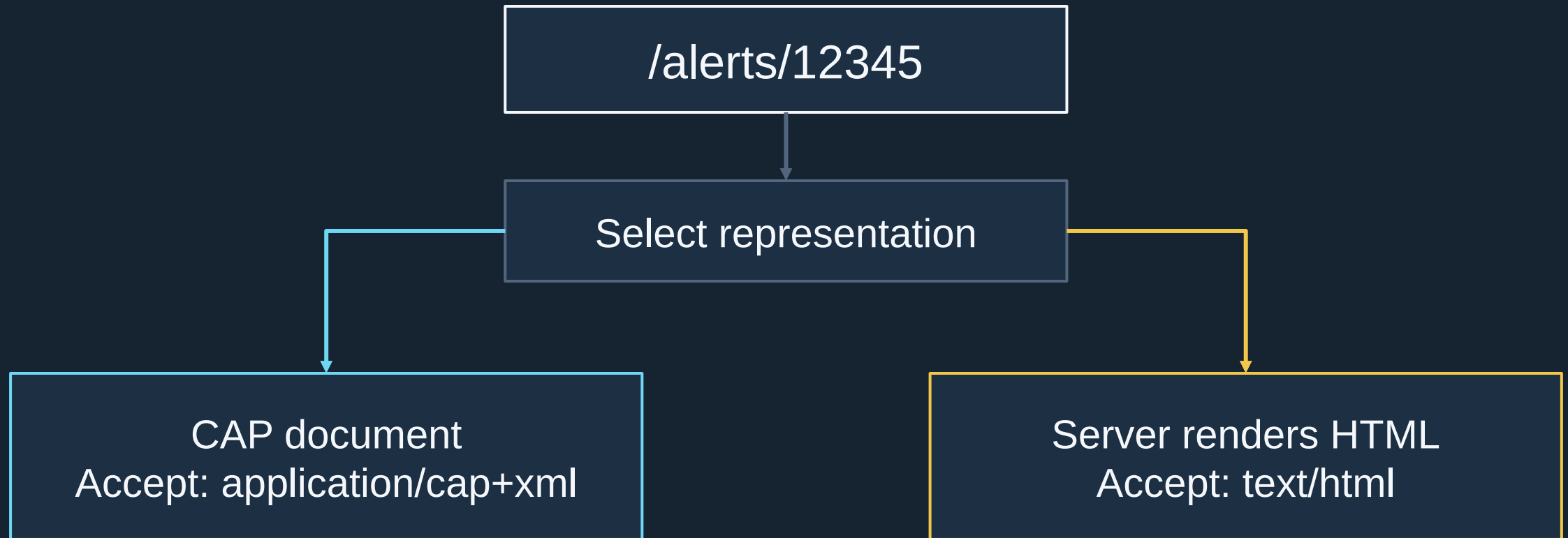
```
GET /alerts/12345
Accept: text/html
```

```
200 OK
Content-Type: text/html
Vary: Accept
```

```
<html ...>
```

Same alert. Same resource. Different representation.

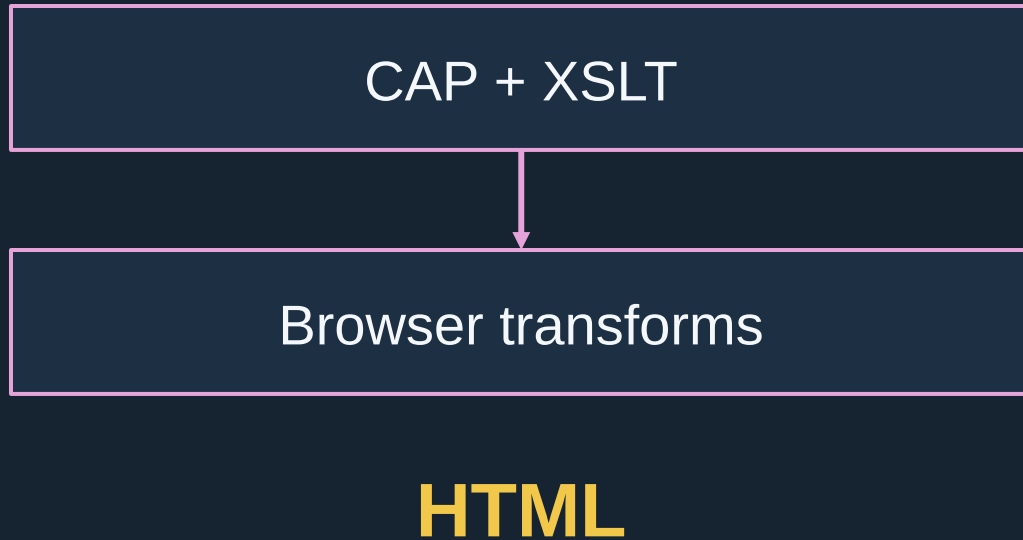
One request reaches one rendering decision



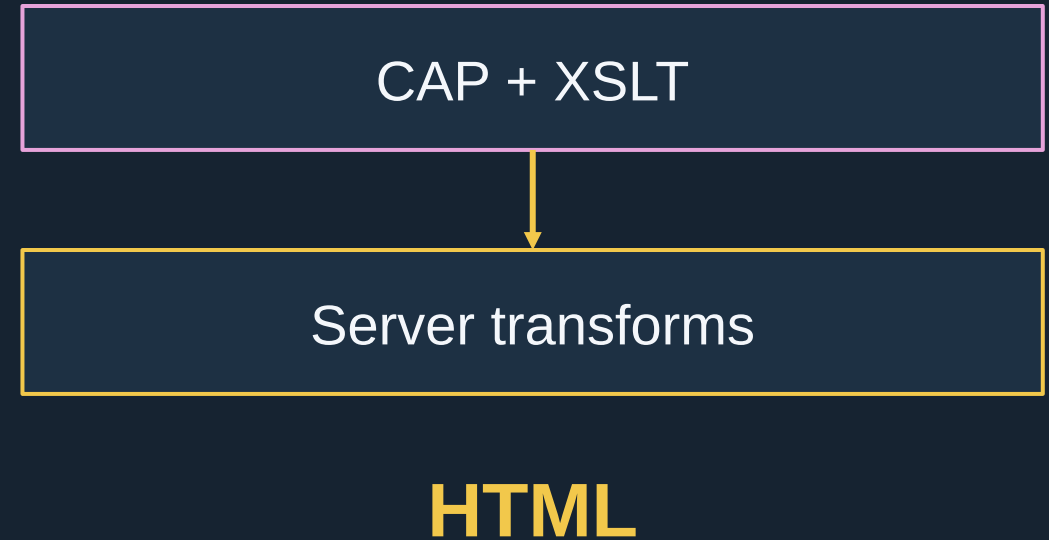
Vary: Accept

Existing XSLT can move to the server

CURRENT

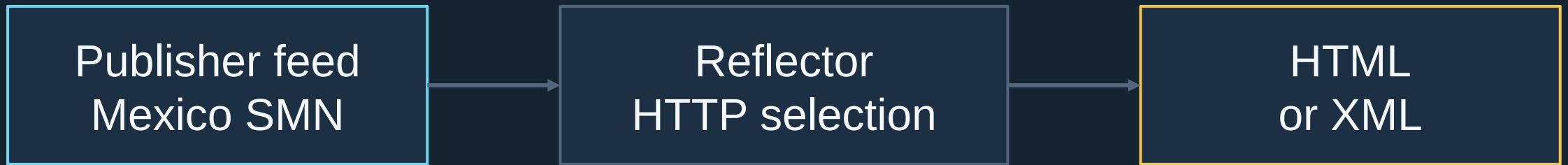


POSSIBLE MIGRATION



Templates and CAP renderers are also valid server options.

One implementation: an Alert-Hub reflector



Feed: HTML or RSS/XML

Individual alert: HTML or CAP XML

Observed 4 Oct 2026: suffixes fix the format.
Accept selects the unsuffixed resource.

Choices depend on the publisher

APPROACH	WHY CHOOSE IT	WHAT TO OWN
CAP/XML only	Simple machine contract	Human view elsewhere
Separate URLs	Explicit formats	Keep views aligned
Client JavaScript	Rich browser interaction	Maintain the client runtime
Server rendering	Conventional HTML	Operate a renderer
Server XSLT	Retain stylesheets	Control execution + trust
Negotiation	One resource, many forms	Define defaults + caching

This is ordinary web development

.NET / ASP.NET Core

Controller output formatters

Java / Spring MVC

produces + message converters

PHP / Symfony

Parsed Accept + typed response

Keep CAP valid. Test the actual HTTP contract.

A practical migration checklist

- 01 Find browser XSLT dependencies
- 02 Define browser responses and machine defaults
- 03 Choose URLs, negotiation and rendering
- 04 Review existing XSLT and the new trust boundary
- 05 Test CAP, HTML, caches and accessibility
- 06 Rehearse failures before browser removal

Include */*, q=0, Vary: Accept and your actual CDN.

CAP remains. Presentation becomes explicit.

CAP

The interoperable alert representation

HTML

The human representation

HTTP

Connects clients to the representation

Decide what each client should receive.

Contact and resources

Ian Ibbotson

CTO, Alert-Hub.org

ian.ibbotson@alert-hub.org

[Telegram: @CAP_community](https://t.me/CAP_community)

blog.alert-hub.org

COMPANION ARTICLE

[CAP after browser-side XSLT](#)

[PowerPoint \(.pptx\)](#)

[PDF presentation](#)

[Rehearsal notes](#)

[Code examples \(.zip\)](#)

The article also includes references and interactive demos.