Google Crawl Infra

@ IETF 123 web-bot-auth BoF

Martin Splitt

Jul 2025

01 Our use-cases

02 Current identifications

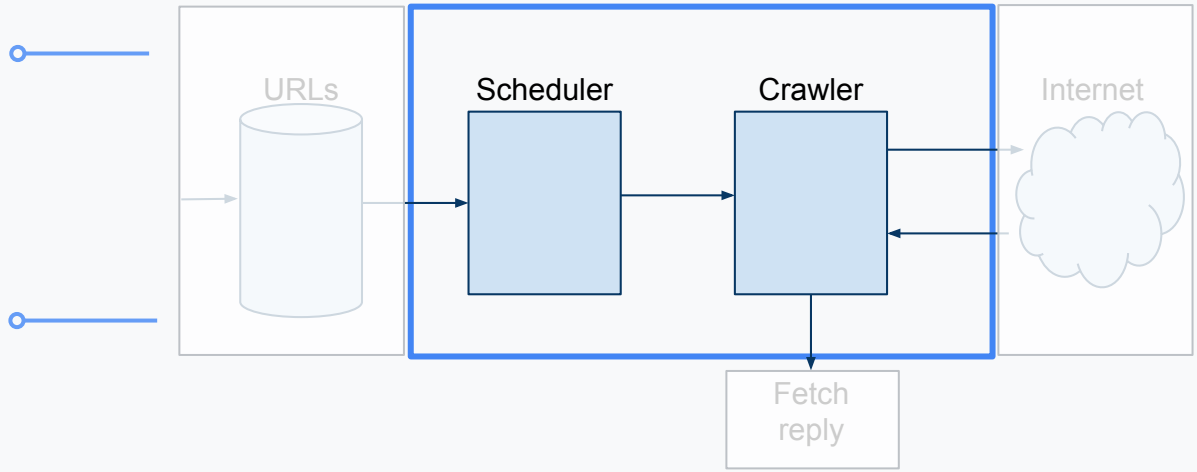
03 Requirements

04 Q&A

Our Crawler & Fetcher infrastructure

Core infrastructure shared across Google to allow enforcing crawling policies (e.g., robots.txt, spam, malware, host load management, prioritization, ...)

Teams configure how their crawler / fetcher identifies itself. For example, Google Search uses "Googlebot", Google Ads uses "AdsBot-Google".



Our thinking on "Crawler" vs "Fetcher"

Crawlers

- fetches for a product / service
- completely automatic (may follow links)
- batch fetches
- very large scale (>billions of URLs)
- aggressive internal caching
- robots.txt obeyed by default

Fetchers

- fetches for user
- user behind each fetch
- single URL fetches
- scale ~1-5 URLs per user
- some caching
- may not use robots.txt

Crawler / Fetcher type samples

Crawlers:

- Web Search crawling ("Googlebot") - for indexing in search, fully automatic, follows links to new content, renders HTML pages (also fetches embedded resources like JS, CSS, XHR, etc), also for images, videos, desktop & mobile user-agents; strong caching: billions+ requests/day
- Ads, shopping - for landing page & metadata review (crawled on submission from site owner)
- AI training - as you do in 2025
- robots.txt checks - as per RFC9309, cached 1 day

Some fetchers:

- Various products - Gmail, Calendar, imports / exports, feeds, etc (mostly on submission)
- Spam / malware checks when users see links
- Live end-user requests - translation, link-previews, read aloud, browser/cloud agents

01 Our use-cases

02 Current identifications

03 Requirements

04 Q&A

Identification via user-agent-string

The request headers are very easily spoofable by clients; even with a standard Google Chrome installation it takes less than a minute to change the client's user-agent to an arbitrary value.

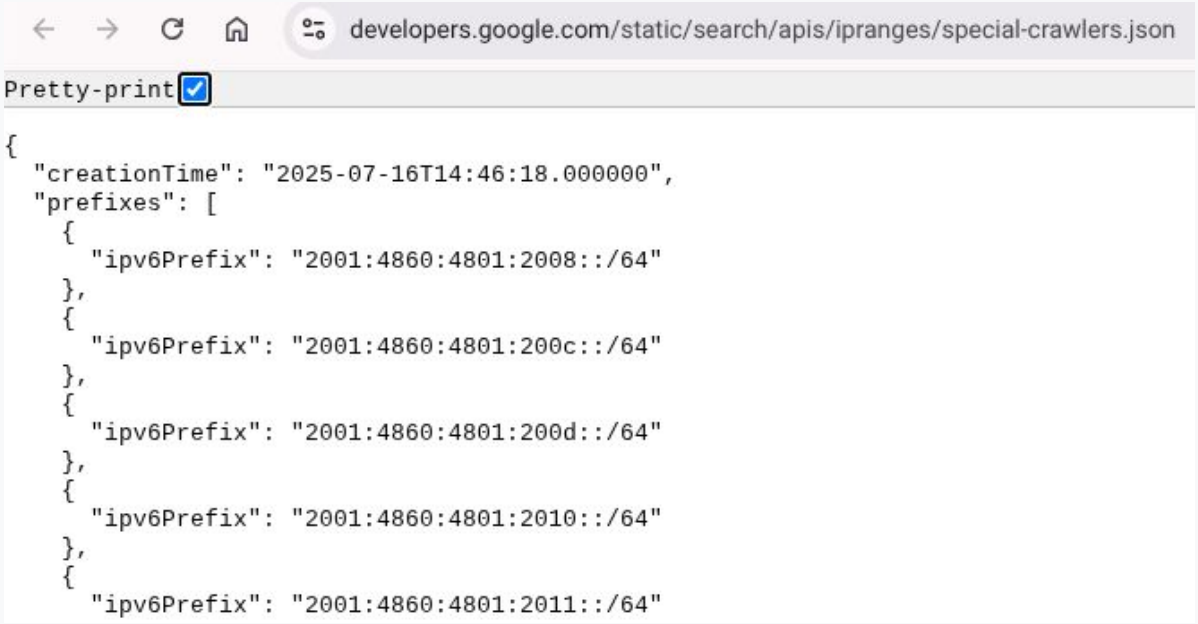
```
Mozilla/5.0 (Linux; Android 6.0.1; Nexus 5X
Build/MMB29P) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/W.X.Y.Z Mobile Safari/537.36 (compatible;
Googlebot/2.1; +http://www.google.com/bot.html)

Mozilla/5.0 AppleWebKit/537.36 (KHTML, like Gecko;
compatible; GoogleOther) Chrome/W.X.Y.Z Safari/537.36

AdsBot-Google (+http://www.google.com/adsbot.html)
```

Identification via IP address

Generally doable, however only viable when the client uses IP ranges dedicated to crawling, i.e. not shared with other clients. This is a seriously hard issue to solve for new crawlers who might operate with very narrow IP range allocations, often sharing that with the whole company or a whole cloud range.



Identification via reverse + forward DNS lookups

Reverse DNS queries against the IPs believed to belong to crawlers is certainly viable (and Google's been providing this data since about 2005), however it's affected by the same issue as the previous point: shared IP ranges. Additionally, new crawlers rarely have management access to the reverse zones of the used IP ranges.

```
$ host 66.249.66.1
1.66.249.66.in-addr.arpa domain name pointer crawl-66-249-66-1.googlebot.com.

$ host crawl-66-249-66-1.googlebot.com
crawl-66-249-66-1.googlebot.com has address 66.249.66.1
```

Example 2:

```
$ host 35.247.243.240
240.243.247.35.in-addr.arpa domain name pointer geo-crawl-35-247-243-240.geo.go

$ host geo-crawl-35-247-243-240.geo.googlebot.com
geo-crawl-35-247-243-240.geo.googlebot.com has address 35.247.243.240
```

01 Our use-cases

02 Current identifications

03 Requirements

04 Q&A

Why does a search engine care about identification?

Enable sites to debug & fix access issues:

- We report crawl failures to site owners. If they want to be crawled, they can use the identification information to work out what went wrong. Did their infrastructure think we were DDOS'ing them & they blocked the IP?
- Live fetcher results (eg, translations) are immediately visible to users - if a site wants to allow them, they may want to make sure that these requests aren't blocked.

Enable sites to confirm it's us:

- When they see requests, they can use identification to confirm it was actually from Google, and use that information to complain to us (eg "you crawl way too much")

Enable sites to deal with fake requests:

- We aim to be a good citizen wrt crawling. Some bots like to piggy-back on our reputation to be treated well too.
- We abide by robots.txt for crawlers; if another bot with our UA doesn't, sites can confirm that it wasn't us.

Identification wish-list & requirements

Simplicity

- **Ease of implementation**: for both sides.
- **Efficient & fast**: The fewer bytes & requests on the wire, the better.
  We crawl a lot. Sites don't like wasted bandwidth or wasted latency.

Scalability

- **For large & small batches**: Works for "mass-requests", and one-offs.
- **Allow temporal disconnect**: Should allow time-delayed identification (eg "last week's request"); within a reasonable timeframe.

Ecosystem-friendliness

- It should be relatively straightforward for the ecosystem to challenge or confirm the authenticity of a client.
- Built on existing standards.

We continue to agree that
crawlers need a robust identification mechanism
and
happy to test solid feasible proposals.

01 Our use-cases

02 Current identifications

03 Requirements

04 Q&A