



LLMs and AI Agents Integration Framework

Proposed framework for publisher content monetization and brand content management for LLMs and AI agents driven consumer interfaces

Released June 4, 2025

Please email support@iabtechlab.com with feedback or questions. This document is available online at <https://iabtechlab.com/llm>

About this Document

The rapid rise of AI agents, large language models (LLMs), and AI-driven search summaries (e.g., Google's Gemini-powered AI Overviews) is transforming how users access information, significantly reducing traffic to publisher websites and apps. This shift, coupled with a surge in AI bot scraping and the concentration of digital advertising revenue among a handful of tech giants, threatens the sustainability of publishers on the open internet. Without intervention, the content ecosystem that fuels AI systems risks collapse, degrading both publisher viability and AI quality.

And it is not just the publishers that will be affected. As AI generated answers solve user information needs, brand reputation and messaging will only be as good as the user's chosen AI system. Thus it is necessary to ensure the AI systems have the right information about the brands and they are all able to get it from the source of truth—the brand itself—in an organized manner.

This document defines a framework and new APIs for

- Publishers and brands to control and provide guided access for LLM and AI agent bots
- Providing LLM friendly content access
- Cost per crawl monetization opportunity and API for interoperability
- Dynamic monetization for content with variable pricing and bidding opportunities for fast moving or in demand content using the LLM Ingest API
- AI native technologies like Model Context Protocol (MCP) based NLWeb protocol to be integrated with LLM Ingest API for publishers and brands

This document is a first step in seeking industry feedback in standardizing the LLM integration framework including the APIs for interoperable management of the new web economy.

License

LLMs and AI Agents Integration document is licensed under a [Creative Commons Attribution 3.0 License](https://creativecommons.org/licenses/by/3.0/). To view a copy of this license, visit creativecommons.org/licenses/by/3.0/ or write to Creative Commons, 171 Second Street, Suite 300, San Francisco, CA 94105, USA.

Author

Shailley Singh, EVP Product & COO, IAB Tech Lab

About IAB Tech Lab

The IAB Technology Laboratory is a nonprofit research and development consortium charged with producing and helping companies implement global industry technical standards and solutions. The goal of the Tech Lab is to reduce friction associated with the digital advertising and marketing supply chain while contributing to the safe growth of an industry.

The IAB Tech Lab spearheads the development of technical standards, creates and maintains a code library to assist in rapid, cost-effective implementation of IAB standards, and establishes a test platform for companies to evaluate the compatibility of their technology solutions with IAB standards, which for 18 years have been the foundation for interoperability and profitable growth in the digital advertising supply chain. Further details about the IAB Technology Lab can be found at <https://iabtechlab.com>.

Disclaimer

THE STANDARDS, THE SPECIFICATIONS, THE MEASUREMENT GUIDELINES, AND ANY OTHER MATERIALS OR SERVICES PROVIDED TO OR USED BY YOU HEREUNDER (THE "PRODUCTS AND SERVICES") ARE PROVIDED "AS IS" AND "AS AVAILABLE," AND IAB TECHNOLOGY LABORATORY, INC. ("TECH LAB") MAKES NO WARRANTY WITH RESPECT TO THE SAME AND HEREBY DISCLAIMS ANY AND ALL EXPRESS, IMPLIED, OR STATUTORY WARRANTIES, INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AVAILABILITY, ERROR-FREE OR UNINTERRUPTED OPERATION, AND ANY WARRANTIES ARISING FROM A COURSE OF DEALING, COURSE OF PERFORMANCE, OR USAGE OF TRADE. TO THE EXTENT THAT TECH LAB MAY NOT AS A MATTER OF APPLICABLE LAW DISCLAIM ANY IMPLIED WARRANTY, THE SCOPE AND DURATION OF SUCH WARRANTY WILL BE THE MINIMUM PERMITTED UNDER SUCH LAW. THE PRODUCTS AND SERVICES DO NOT CONSTITUTE BUSINESS OR LEGAL ADVICE. TECH LAB DOES NOT WARRANT THAT THE PRODUCTS AND SERVICES PROVIDED TO OR USED BY YOU HEREUNDER SHALL CAUSE YOU AND/OR YOUR PRODUCTS OR SERVICES TO BE IN COMPLIANCE WITH ANY APPLICABLE LAWS, REGULATIONS, OR SELF-REGULATORY FRAMEWORKS, AND YOU ARE SOLELY RESPONSIBLE FOR COMPLIANCE WITH THE SAME, INCLUDING, BUT NOT LIMITED TO, DATA PROTECTION LAWS, SUCH AS THE PERSONAL INFORMATION PROTECTION AND ELECTRONIC DOCUMENTS ACT (CANADA), THE DATA PROTECTION DIRECTIVE (EU), THE E-PRIVACY DIRECTIVE (EU), THE GENERAL DATA PROTECTION REGULATION (EU), AND THE E-PRIVACY REGULATION (EU) AS AND WHEN THEY BECOME EFFECTIVE.

Glossary

AI	Artificial intelligence (AI) refers to the ability of machines to perform tasks that typically require human intelligence, such as learning, problem-solving, and decision-making. In essence, AI aims to create systems that can simulate human intelligence and cognitive abilities.
AI Agents	An AI agent is a software program designed to interact with its environment, process information, and take actions to achieve specific goals. Unlike traditional AI models, which are often static and trained for specific tasks, AI agents are dynamic and can learn, adapt, and make decisions based on their surroundings and interactions.
API key	An API (Application Programming Interface) key is a unique identifier, often a string of characters, used to authenticate and authorize a user, application, or calling program to an API. It acts like a password, allowing access to an API's functionality. API keys are crucial for security, identifying the source of API requests, and managing access.
Bots	A bot, short for robot, is a software application that performs automated tasks on the internet. These tasks can be anything from indexing websites (like search engine bots) to simulating human interaction (like chatbots). Bots can also be malicious, used for spamming, cyberattacks, or other harmful activities.
CDN	A CDN (Content Delivery Network) is a network of geographically distributed servers that work together to quickly deliver web content to users around the world. By caching and serving content from servers closer to users, CDNs significantly reduce latency and improve website load times.
chatbots	A chatbot is a computer program designed to simulate human conversation, typically through text or voice interfaces. They can range from simple rule-based systems to more sophisticated AI-powered models that use natural language processing (NLP) to understand and respond to user queries.
CPCr	Cost per Crawl or it represents the cost of crawling a site 1000 times
JSON	JSON (JavaScript Object Notation) is a lightweight, human-readable text-based format for representing structured data. It's commonly used for transmitting data between a server and a web application, and it's also used for storing data in files or databases. JSON is designed to be easy to read and write by humans, and it's also easily parsed by machines.

LLM	A Large Language Model (LLM) is a type of artificial intelligence that excels at understanding and generating human language. It's trained on massive datasets of text, allowing it to learn patterns and relationships in language, enabling it to perform various tasks like text generation, translation, summarization, and answering questions.
MCP	MCP stands for Model Context Protocol. It's a protocol designed to standardize how AI models, like LLMs, interact with external tools and data sources. Think of it as a universal connector for AI, allowing models to access information, use APIs, and perform actions beyond their inherent knowledge.
Meta tags	Meta tags are pieces of code within an HTML document's <head> section that provide information about the web page, primarily used to describe the page's content and its purpose. This data, known as metadata, is not displayed on the webpage itself but is used by search engines, browsers, and other web services.
NLWeb	NLWeb is an open-source project spearheaded by Microsoft, aiming to bring conversational AI capabilities directly to websites.
RAG	A Retrieval-Augmented Generation (RAG) model is a technique that enhances the performance of Large Language Models (LLMs) by integrating them with a mechanism for retrieving and incorporating external data. RAG models essentially combine the generative capabilities of LLMs with the ability to access and utilize information beyond their training data.
Robots.txt	A robots.txt file is a plain text file that provides instructions to web crawlers, such as search engine bots, about how to access and index a website. It's essentially a set of rules that website owners can use to control which pages or sections of their site are crawled and indexed by search engines.
RSS	RSS, often expanded as Really Simple Syndication, is a web feed format used to distribute content like blog posts, news updates, and podcasts in a standardized format. It allows users to subscribe to updates from their favorite websites and view them in a centralized location using an RSS reader. Essentially, RSS acts as a shortcut to stay informed about updates from various sources without having to visit each website individually.
Schema.org	Schema.org is a collaborative effort, initiated by Google, Microsoft, Yahoo, and Yandex, to create a vocabulary of terms that help search engines understand the context of content on web pages. It's a standardized way to structure structured data using markup (like HTML) so that machines can interpret the information more effectively. Think of it as a "lingua franca" for

search engines, helping them understand what a page is about and how to display it in results.

Token

LLMs don't process raw text directly. They first break down the text into smaller, manageable units called tokens. Tokens are the basic building blocks that LLMs use to understand and generate text. It's essentially a chunk of text that the LLM uses as input or output during both training and inference (text generation). Tokens can be entire words, parts of words, punctuation marks, or even spaces.

User Agents

A user agent is a computer program representing a person, for example, a browser in a Web context. Besides a browser, a user agent could be a bot scraping webpages, a download manager, or another app accessing the Web. Along with each request they make to the server, browsers include a self-identifying User-Agent HTTP header called a user agent (UA) string. This string often identifies the browser, its version number, and its host operating system. Spam bots, download managers, and some browsers often send a fake UA string to announce themselves as a different client

Web Application Firewall (WAF)

A Web Application Firewall (WAF) is a type of firewall specifically designed to protect web applications and APIs from attacks that exploit vulnerabilities in their logic and code. It operates by filtering and monitoring HTTP/HTTPS traffic, inspecting requests for malicious patterns, and blocking or alerting on suspicious activity.

Table of Contents

About this Document	2
Glossary	4
The AI Challenge and Opportunity	9
Publisher Problem	9
Brand Problem	10
Summary	11
Publisher Framework	13
Manage and control access to content	13
Provide LLM friendly discovery	13
Monetize content	13
Brand Framework	15
Understand access to content	15
Provide LLM friendly discovery	15
LLM ingest API	15
Content access controls	16
Robots.txt	16
Web Application Firewall (WAF) Methods	16
Redirect to Access Controls Rules	17
Provide LLM Friendly Discovery	19
Content Access Rules Page	19
Content metadata	20
llms.txt	20
Publisher Cost per Crawl (CPCr) framework	23
Discovery	24
CPCr API	25
Publisher LLM Ingest API	31
Authentication	31
Endpoints	31
Discovery Endpoint	31
Query Endpoint	33
Bidding Endpoint	35
Logging Endpoint	37
Pricing Options	38
Content Access	39
Error Responses	39
Example Requests	39
Brand LLM Ingest API	42
Authentication	42

Endpoints	42
Discovery Endpoint	42
Query Endpoint	44
Logging Endpoint	47
Content Access	48
Example Requests	49
NLWeb option for publishers and brands	52
NLWeb vs LLM Ingest API	52
Recommendations for using NLWeb option	54
Conclusion	58

The AI Challenge and Opportunity

AI agents, LLMs, and AI-driven search summaries are reducing publisher traffic by 20–60% (up to 90% for niche sites), with an estimated \$2 billion ad revenue loss.

TollBit's report shows AI's impact with 117% bot traffic surge (5.05M scrapes/site, 1.89M hidden scrapes) and 95.7% fewer referrals (0.37% vs. 8.63% Google). At the same time 80.8% ad revenue is shared by the top 10 and 11.0% by the top 11–25. This highlights the trend in ad spend concentration among top technology platforms, squeezing open internet publishers. AI companies are valued at billions, yet they rely on open internet content without compensating content owners. This model risks the entire ecosystem.

Publisher Problem

Traffic Declines from AI-Driven Search

AI-driven search summaries, such as Google's Gemini-powered AI Overviews, deliver instant answers, reducing publisher website traffic by

- **20–60%** on average, with niche sites (e.g., travel, food, DIY) experiencing losses up to **90%** as per ecommercegermany.com, emarketer.com.
- Referral traffic from Google search has declined by **45%** in some cases mi-3.com.au.
- The ratio for pages scraped to site visits has fallen from 2:1 to 6:1 for Google search but is more dramatic for LLMs at 250:1 for Chat GPT and 6000:1 for Claude as per [Matthew Prince, Cloudflare CEO](https://www.cloudflare.com/insights/ai-scraping)
- AI chatbots drive **95.7% fewer click-throughs** compared to traditional search, with a referral rate of **0.37%** versus **8.63%** for Google's search engine results page (SERP) as per [TollBit State of the Bots report Q4 2024](https://tollbit.com/state-of-the-bots-report-q4-2024)

Renowned travel blog The Planet D, reported a 90% drop in traffic during Gemini's initial rollout, [also according to t3n](https://www.t3n.com). The couple had to lay off staff and ultimately shut down their website. "I feel betrayed," Bouskill said. "We built something for 16 years, and it was gone in six months."

Surge in AI Bot Scraping

Bot traffic that scrapes and ingests content from publisher websites has seen a surge as a result of growth of LLMs and AI agents.

- Bot traffic accounts for 51% of web activity, with bad bot traffic rising from 32% to 37% annually, [Infosecurity Magazine](https://www.infosecuritymagazine.com).
- TollBit's Q4 2024 report shows a 117% QoQ increase in AI bot traffic, with sites scraped 5.05 million times on average (7.2 scrapes/page) as per [TollBit State of the Bots report Q4 2024](https://tollbit.com/state-of-the-bots-report-q4-2024). Hidden scraping (1.89 million scrapes/site) nearly matches identified AI bot scrapes (2 million/site)

- Leading bots include ChatGPT-User (15.6% of AI traffic, 6,767.6% growth), Bytespider (12.44%), and ClaudeBot (566.79% growth) as per TollBit Q4 2024 report.
- 3.3% of scrapes bypass robots.txt, with a 40% increase in unauthorized scraping from Q3 to Q4.

The surge in bot scraping, often for LLM training or real-time RAG, burdens publishers with rising cybersecurity costs without compensation for scraped content.

Revenue Losses and Concentration

- Reduced traffic translates to an **estimated \$2 billion annual ad revenue loss** for the publishing industry [Digiday](#).
- The [IAB/PwC 2024 report](#) shows internet advertising grew **14.9%**, but **80.8%** of revenue is concentrated among the **top 10 companies** (e.g., Google, Meta), with the **top 11–25** (social media, streaming, e-commerce) holding **11.0%**, up **3.1%** from Search advertising (**\$102.9 billion, 15.9% growth**), programmatic (**\$134.8 billion, 18.0% growth**), social media (**\$88.8 billion, 36.7% growth**), and retail media networks (**\$53.7 billion, 23.0% growth**) dominate ad spend.

The concentration of ad revenue among a few tech giants limits monetization opportunities for smaller publishers, exacerbated by AI-driven search trends

Ethical and Economic Disparities

AI companies, valued at billions, profit by using publisher content for training and real-time RAG without payment or attribution. Unauthorized scraping, evidenced by hidden bots and a recent [lawsuit from Dow Jones](#), and [40% robots.txt violations](#) reflect unethical practices. Publishers nor Brands are rarely informed about content usage, deepening the economic imbalance.

AI Dependency on Publisher Content

LLMs rely on fresh, human-generated content to maintain accuracy and avoid hallucinations [TechPolicy.Press](#). TollBit report's 2 million scrapes/site and 117% AI bot traffic surge demonstrate AI's heavy dependence on publisher content.

Brand Problem

As more consumers rely on LLM based chat bots and AI agents for their work and information needs, there is risk for brand reputation.

Understanding User Intent

- LLMs rely on prompts, and without access to these prompts, brands struggle to understand what users are asking about and what their needs are.
- This makes it difficult to tailor content and brand messaging to effectively address user queries.

Content and Messaging Alignment

LLMs can generate text that may not align with a brand's message or values.

- This can lead to a mismatch between what users expect and what they receive, potentially damaging brand perception.

Evolving LLM Outputs

- LLMs can produce different responses to the same prompts over time, even if the underlying data doesn't change.
- This requires continuous monitoring and adaptation to ensure brand consistency and accuracy.

Brand Reputation Risks

- Hallucinations: LLMs can generate false or nonsensical information, which can damage a brand's reputation if not carefully managed.
- Misrepresentation of Brand Values: AI can sometimes misinterpret or misrepresent a brand's values, leading to negative perceptions.

Summary

The traditional business mode of the web publisher was:

Search engine scrapes websites >> User searches for information>> Search Engine sends user to publisher and brand website>> Website shows their content and earns revenue through ads

This model is being upended as bots like ChatGPT-User (**15.6%** of traffic) generate no or minimal ad revenue for publishers as they don't view ads or subscribe to content and drop the site referral by up to 90%.

The new model is

AI engine crawls content >> Users deploy AI agents for information>> AI Driven Results (AIDR) present publisher and brand content to users (no or highly reduced direct content/ website/ brand interaction)>> AI agent shows ads and earns revenue

Publisher Opportunity

At the same time, a surge in scraping activity by AI based bots shows that LLMs and AI agents continuously need new content. Thus there is value in content and opportunity for publishers to monetize content by seeking fair compensation from AI and search bots. There is an opportunity for publishers to optimize for AI and license their content.

Brand Opportunity

Similarly for brands there is a need to optimize for AI and develop collaboration with LLMs and AI agents to understand the prompts better and provide true and relevant content for use in AI based applications.

Publisher Framework

Some publishers have signed deals with LLM providers for e.g. Open AI to provide content in exchange for financial compensation. That is a great start but that may not truly value content at market rates. It may work for larger publishers with a critical mass of content and systems to support such transactions.

The web is a much more diverse place with publishers of all sizes and variety of content. Some content may be more valuable at a certain point in time than others and the new business model should be able to value the content according to the need and urgency.

In order to monetize content in the world of AI bots, we propose the following three step approach for publishers. Each step is described in more detail, in later sections with guidance and suggested standard methods and draft API definitions.

Manage and control access to content

Prevent bots and scrapers from accessing content deploying access control mechanism like

- **Robots.txt** declarations
- Web Application Firewall methods (**WAF**)

Provide LLM friendly discovery

- Develop and deploy mechanisms that enable AI systems to easily discover and understand publisher content
- **Content Access Rules** page
- **llms.txt file** describing content in markdown llm friendly language
- **Content meta data** file

Monetize content

Different revenue models can be adopted by publishers to enable buying and selling of content. There are different revenue models with which publishers and content providers can engage LLMs and AI agents to monetize their content. Choice of revenue model can depend on the volume and nature of content as well as ease of doing business. Publishers may choose one or a combination of several business models.

- **Content Partnerships:** Already being done by many large content providers with multi year content libraries. These can be enabled by providing access to content via well defined API. These are effective for publishers with large content libraries and access to historic content.

- **Cost per crawl:** Publishers can also set a price each time an LLM or an AI agent crawls their website. This is an easy way to monetize their website content with minimal changes. There are platforms today providing such capabilities that enable selective blocking for bots and agents as well as logging and measurement for billing purposes.
- **LLM ingest content API:** Publishers can also price their content based on demand and deploy API access for LLMs and agents to query specific content based on user prompts. The LLM or agent can submit a request for content with the price per token they are willing to pay and the publisher can honor the price and return results with content and /or authorization code and paths to the content. Or the publisher can refuse the request. Publishers can price the content in the content discovery guidance, This method allows publishers to value content based on recency and demand for specific content categories.

Brand Framework

Brands' objective is to protect their reputation and ensure that LLMs and AI agents have the correct information that aligns with their brand value and market message as well as provides relevant information about their product and services.

Brands need answers to the following questions:

- Who is accessing their content
- How many times is their content accessed
- What kind of queries or user prompts are triggers for content access

For brands we recommend the following two steps to direct the LLMs and AI agents to the right content.

Understand access to content

Manage permissions to content access and monitor content access using:

- **Robots.txt** declarations to control/ guide access
- Web Application Firewall (**WAF**) to track and monitor

Provide LLM friendly discovery

Develop and deploy mechanisms that enable AI systems to easily discover and understand publisher content

- **llms.txt file** describing content in markdown llm friendly language
- **Content meta data** file

LLM ingest API

Provide a mechanism to receive queries and user prompts so the brand can provide relevant and correct content. They can use the LLM ingest API described later [here](#) without the pricing controls

***** Please Note: LLMs and AI agents may collect or receive information about a brand and their products from sources other than brand owned content. This document only covers the content owed by the brand and DOES NOT address mechanisms to manage content not owned by the brand. *****

Content access controls

To prevent bots and scrapers from accessing your content freely, there are several mechanisms available and publishers may deploy one or more of these to make them more effective.

Robots.txt

- **Description:** A text file placed at the root of your site (e.g., <https://your-site.com/robots.txt>) to instruct web crawlers on what they can or cannot access.
- **Implementation:** Disallow specific bots or all bots from accessing content.

Unset

```
User-agent: GPTBot
Disallow: /
User-agent: *
Disallow: /premium-content/
```

Pros: Simple, widely recognized by compliant bots.

Cons: Non-compliant bots (e.g., malicious scrapers) may ignore it.

Web Application Firewall (WAF) Methods

Web Application Firewalls (WAFs) are powerful tools designed to enhance the security of web applications by filtering and controlling incoming traffic. They are particularly effective for managing bot traffic, protecting content, and enforcing access rules. Key WAF Methods for Security and Bot Management

- **Bot Detection and Blocking:** WAFs can identify and block bots by analyzing various indicators, such as:
 - **IP Addresses:** Block traffic from known bot-associated IPs or ranges.
 - **User Agents:** Deny requests from specific bot identifiers (e.g., GPTBot, ClaudeBot).
 - **Request Patterns:** Flag rapid or repetitive requests typical of bots.
 - **Behavioral Analysis:** Use machine learning to detect unusual activity, like non-human navigation patterns.
- **Rate Limiting:** WAFs restrict the number of requests a single IP or user agent can make within a set time frame.

- **Custom Rules:** WAFs allow you to define tailored rules to block specific bots or traffic patterns for e.g. a rule to block requests with the user agent GPTBot or from a known scraper IP range.
- **Challenge Mechanisms:** WAFs can present challenges to verify if a visitor is human, such as:
- **Logging and Monitoring:** WAFs provide detailed logs of traffic and bot activity, enabling you to track patterns and refine rules over time.
- **Geo-Blocking:** WAFs can block traffic from specific regions known for high bot activity.
- **Content Scraping Prevention:** WAFs detect and stop scraping by monitoring for rapid, repetitive requests or abnormal navigation patterns.
- **Redirection to Rules Page:** Instead of simply denying access, WAFs can redirect blocked bots to a page explaining content access rules. This educates bot operators on legitimate access options (e.g., API signup).

Redirect to Access Controls Rules

When bots are blocked or detected, guide them to a standardized page (e.g., `/content-access-rules`) that outlines how to access your content legitimately. Here are several methods that a publisher can use to redirect for content access rules:

Custom Error Pages

Serve a 403 Forbidden page with a link to the rules page when a bot is blocked and configure your server to redirect blocked requests.

Unset

```
<h1>Access Denied</h1>
<p>Review our <a href="/content-access-rules">Content Access
Rules</a> for legitimate access.</p>
```

HTTP Headers

Add a custom header to responses, pointing to the rules page to be included in server responses.

Unset

```
X-Content-Rules: https://your-site.com/content-access-rules
```

Pros: Machine-readable for compliant bots.

Meta Tags

Embed metadata in HTML to signal the rules page and add to your pages' <head> section.

Unset

```
User-agent: *
```

```
Disallow: /
```

```
# Visit https://your-site.com/content-access-rules for access rules
```

Pros: Visible to crawlers parsing HTML.

Robots.txt Extension

Add a custom directive or comment in robots.txt linking to the rules page.

Unset

```
User-agent: *
```

```
Disallow: /
```

```
# Visit https://your-site.com/content-access-rules for access rules
```

Pros: Leverages an existing standard.

Cons: Not all bots will read comments.

Provide LLM Friendly Discovery

Next step after detecting and blocking bots is to guide them to the content access rules where publishers can provide discovery i.e. details about the available content, how to access the content and any other details or instructions LLMs and AI agents need to follow to access publisher content.

Content Access Rules Page

The purpose of the content access rules is to inform the access rules and where to find more information. Suggested structure to include following minimum information

- Terms of Use: Rules for accessing content (e.g., no scraping without permission).
- How to Access: Instructions for API keys, subscriptions, or licensing.
- Contact Info: Email or form for inquiries.
- Legal Notices: Copyright and consequences of violations
- Structured meta data about the content

Unset

```
<h1>Content Access Rules</h1>
<p>Unauthorized scraping is prohibited</p>
<ul>
  <li>Legal copyright notice is available at <a href="/terms of use"></a>.</li>
</ul>
<p>To access our content:</p>
<ul>
  <li>Request an API key at <a href="/api/signup">/api/signup</a>.</li>
  <li>Contact licensing@your-site.com for custom agreements.</li>
</ul>
<p>Our content is described in structured formats for AI systems </p>
<ul>
  <li> Access metadata at <a href="/content-metadata.json">/content-metadata.json</a>. </li>
  <li> Access llms.txt at <a href="/llms.txt">/llms.txt</a>. </li>
</ul>
```

There are two ways to present the structured meta data about the content. The traditional method is to publish the content meta data in JSON format. Another method is recently developed is llms.txt - markdown file for describing content categories and more about the website content.

Content metadata

This file is used to inform the LLMs about different categories of content published by the publisher.

```
Unset
{
  "content": [
    {
      "domain": "examplesports.com",
      "summary": "news and information about sports",
      "keywords": ["sports," "news," "apparel," "equipment
    }
  ]
  { "cat": [
    "483", // sports
    "483-521", // Olympic sports
    "483-512" // Golf
  ]
}
}
```

llms.txt

A markdown file to websites to provide LLM-friendly content. This file offers brief background information, guidance, and links to detailed markdown files. More details about llms.txt is available here: <https://github.com/AnswerDotAI/llms-txt> . Here is an example llms.txt

```
Unset
# FastHTML

> FastHTML is a python library which brings together Starlette,
Uvicorn, HTMX, and fastcore's `FT` "FastTags" into a library for
creating server-rendered hypermedia applications.
```

Important notes:

- Although parts of its API are inspired by FastAPI, it is **not** compatible with FastAPI syntax and is not targeted at creating API services
- FastHTML is compatible with JS-native web components and any vanilla JS library, but not with React, Vue, or Svelte.

Docs

- [FastHTML quick start](https://answerdotai.github.io/fasthtml/tutorials/quickstart_for_web_devs.html.md): A brief overview of many FastHTML features
- [HTMX reference](https://raw.githubusercontent.com/path/reference.md): Brief description of all HTMX attributes, CSS classes, headers, events, extensions, js lib methods, and config options

Examples

- [Todo list application](https://raw.githubusercontent.com/path/adv_app.py): Detailed walk-thru of a complete CRUD app in FastHTML showing idiomatic use of FastHTML and HTMX patterns.

Optional

- [Starlette full

documentation](<https://gist.githubusercontent.com/path/starlette-sml.md>): A subset of the Starlette documentation useful for FastHTML development.

To create effective llms.txt files, consider these guidelines:

- Use concise, clear language.
- When linking to resources, include brief, informative descriptions.
- Avoid ambiguous terms or unexplained jargon.
- Run a tool that expands your llms.txt file into an LLM context file and test a number of language models to see if they can answer questions about your content

Publisher Cost per Crawl (CPCr) framework

Cost per crawl method of monetization allows publishers to monetize their content when crawled by AI bots (e.g., LLMs, web scrapers, or AI agents). This framework:

- Provides a transparent, auditable system for tracking and billing crawls.
- Ensure secure access to content while preventing abuse or unauthorized crawling.
- Create a scalable and flexible pricing model that accounts for content value, crawl frequency, and bot type.

It may be done by the publisher or an intermediary platform provider that manages content access capabilities on behalf of the publisher. Following capabilities need to be enabled.

Bot Authentication

- Implement an API Key or Token-based system for bots. Each AI bot operator registers with the publisher or a centralized CPCr platform and receives a unique key.
- Use OAuth 2.0 or similar protocols for secure authentication.
- Optionally, integrate with a robots.txt extension to signal CPCr-enabled endpoints (e.g., `Crawl-Cost: /api/cpc/v1`).

Access Control

- Publishers host content behind a paywall-like gateway for bots, requiring authentication before serving content. Alternatively a platform can help publishers manage the controls.
- Use IP whitelisting or domain-based verification to restrict access to verified AI bot operators.
- Implement rate-limiting to prevent excessive crawling (e.g., max 100 requests/min per key).

To accurately bill for crawls, the system must log each interaction:

Crawl Metadata

- Log: Bot ID (via API key), timestamp, URL accessed, content type (e.g., text, image, video), response size (in bytes), and HTTP status code.
- Use a unique request ID for each crawl to enable auditing and dispute resolution.

Logging Infrastructure

- Deploy a server-side logging system to store crawl data in real-time.
- Use a Content Delivery Network (CDN) with logging capabilities (e.g., Cloudflare, Akamai, Fastly) to track requests at scale.

Crawl Verification

- Publishers provide a logging endpoint (e.g., /cpc/log) where bots report their activity for cross-verification.
- Use digital signatures to validate log integrity.

The pricing model determines how publishers charge for crawls:

Cost Per Crawl (CPCr)

- Base rate: Charge per HTTP request (e.g., \$0.001 per request).
- Tiered pricing based on:
 - Content Type: Higher rates for premium content (e.g., articles, research papers) vs. public pages (e.g., blog posts).
 - Content Volume: Charge based on data size (e.g., \$0.01 per MB of content served).
 - Bot Type: Lower rates for academic/research bots, higher for commercial LLMs.
 - Crawl Frequency: Discounts for high-volume crawlers or subscriptions (e.g., \$100/month for unlimited crawls).
 - Example: A news article crawl might cost \$0.005, while a 10MB dataset crawl costs \$0.10.

Dynamic Pricing

- Use a bidding system where publishers set minimum CPCr rates, and bot operators bid based on content value.
- Adjust rates based on demand (e.g., trending news articles cost more during peak events).

Free Tier

Allow limited free crawls (e.g., 100 requests/month) to attract smaller bot operators or

Discovery

Discovery of CPCr monetization can be done in simple formats like robots.txt file as follows:

Unset

User-agent: *

CPC-Discovery: <https://publisher.com/cpc/info>

CPC-Cost: 0.001

CPC-Registration: <https://publisher.com/cpc/register>

OR the content access rules human-readable page (e.g., `/cpc/info`) with embedded JSON-LD for machine readability:

Unset

```
<html>
  <head>
    <script type="application/ld+json">
      { /* JSON structure as above */ }
    </script>
  </head>
  <body>
    <h1>CPCr Pricing and Access</h1>
    <p>Base rate: $0.001 per crawl</p>
    <p>Register: <a href="/cpc/register">Here</a></p>
  </body>
</html>
```

CPCr API

Alternatively publisher platforms can implement an API for bots to integrate with publishers for more comprehensive content ingestion. This section describes the API for discovering and registering for Cost Per Crawl (CPCr) details.

General Metadata

- **CPCr Version:** string (e.g., "1.0") - The version of the CPCr API.
- **Publisher:**
 - **Name:** string (e.g., "Example News") - Human-readable name of the content publisher.
 - **ID:** string (e.g., "pub_12345") - Unique identifier for the publisher.
 - **Contact:** string (Email or URL) - Email address or URL for bot operator support or registration.

Authentication

- **Method:** string (e.g., "api_key", "OAuth2") - The authentication method required to access the API.
- **Registration URL:** string (URL) - URL where bot credentials can be obtained.
- **Scopes:** array (e.g., ["read_content", "read_metadata"]) - Permissions required to access different parts of the API.
- **Rate Limits**
 - **Requests Per Minute:** integer (e.g., 100) - Maximum number of requests allowed per minute.
 - **Requests Per Day:** integer (e.g., 10000) - Maximum number of requests allowed per day.

Pricing

- **Currency:** string (e.g., "USD", "ETH") - The currency used for pricing.
- **Base Rate:** number (e.g., 0.001) - The default cost per content crawl.
- **Tiers:** An array of pricing tiers based on content type. Each tier includes:
 - **Content Type:** string (e.g., "article", "image", "video") - The type of content.
 - **Rate:** number (e.g., 0.005) - The cost per crawl for this content type.
 - **Description:** string (e.g., "Full-text news articles") - A human-readable explanation of the tier.
 - **Size Limit:** string (e.g., "10MB") - Maximum size of content allowed for this tier.
 - **Restrictions:** array (e.g., ["no_archival", "no_commercial_use"]) - Usage restrictions for this content type.
- **Subscriptions:** An array of subscription plans. Each plan includes:
 - **Plan ID:** string (e.g., "basic") - Unique identifier for the subscription plan.
 - **Name:** string (e.g., "Basic Plan") - Human-readable name of the subscription plan.
 - **Cost:** number (e.g., 100) - The cost of the subscription.
 - **Period:** string (e.g., "monthly", "yearly") - The billing frequency of the subscription.
 - **Limits:** string (e.g., "100000 requests or 1GB data") - Usage limits included in the subscription.
 - **Description:** string (Plan details) - Detailed information about the subscription plan.
- **Dynamic Pricing:**
 - **Enabled:** boolean - Indicates if dynamic pricing is enabled.
 - **Bidding Endpoint:** string (URL, e.g., "/cpc/bid") - URL where bids for

content crawls can be submitted.

- **Minimum Bid:** number (e.g., 0.002) - The lowest acceptable bid for a content crawl.

Content Metadata

- **Categories:** array (e.g., ["news", "opinion", "research"]) - Categories of content available.
- **Formats:** array (e.g., ["text", "json", "html", "image"]) - Formats in which content is available.
- **Endpoints:** An array of API endpoints for accessing content. Each endpoint includes:
 - **Path:** string (e.g., "/api/cpc/v1/content") - The URL path for the endpoint.
 - **Description:** string (Purpose of the endpoint) - Explains what this endpoint is used for.
 - **Methods:** array (e.g., ["GET"]) - HTTP methods supported by this endpoint.
 - **Parameters:** object (e.g., { "id": "string", "format": "string" }) - Parameters that can be used with this endpoint (name: type, description). For example:
 - ``id``: string - The unique identifier of the content.
 - ``format``: string - The desired format of the content.
- **Restrictions:**
 - **Geo:** array (e.g., ["US", "EU"]) - Geographic regions where content access is allowed.
 - **Bot Types:** array (e.g., ["commercial", "academic"]) - Types of bots allowed to access the content.

Logging Requirements:

- **Endpoint:** string (URL, e.g., "/cpc/log") - URL where crawl logs should be submitted.
- **Required Fields:** array (e.g., ["bot_id", "url", "timestamp", "size"]) - Data fields that must be included in crawl logs.
- **Signature Method:** string (e.g., "HMAC-SHA256") - Method used to sign crawl log submissions.

Terms and Conditions:

- **URL:** string (Link to the CPCr terms of service) - URL of the terms of service document.

- **Last Updated:** string (e.g., "2025-05-01") - Date when the terms of service were last updated.

Example Interaction for CPCr

Bot Queries Discovery Endpoint:

bash

```
Unset  
GET https://publisher.com/api/cpc/v1/info
```

Receives the JSON response

```
Unset  
{  
  "cpc_version": "1.0",  
  "publisher": {  
    "name": "Example News",  
    "id": "pub_12345",  
    "contact": "cpc@example.com"  
  },  
  "authentication": {  
    "method": "api_key",  
    "registration_url": "https://publisher.com/cpc/register",  
    "scopes": ["read_content", "read_metadata"],  
    "rate_limits": {  
      "requests_per_minute": 100,  
      "requests_per_day": 10000  
    }  
  },  
  "pricing": {  
    "currency": "USD",  
    "base_rate": 0.001,  
    "tiers": [  
      {  
        "content_type": "article",  
        "rate": 0.005,  
        "description": "Full-text news articles",  
        "size_limit": "10MB",
```

```
    "restrictions": ["no_archival"]
  },
  {
    "content_type": "image",
    "rate": 0.01,
    "description": "High-resolution images",
    "size_limit": "5MB",
    "restrictions": ["no_commercial_use"]
  }
],
"subscriptions": [
  {
    "plan_id": "basic",
    "name": "Basic Plan",
    "cost": 100,
    "period": "monthly",
    "limits": {
      "requests": 100000,
      "data_volume": "1GB"
    },
    "description": "Unlimited crawls up to 100K requests/month"
  }
],
"dynamic_pricing": {
  "enabled": true,
  "bidding_endpoint": "https://publisher.com/cpc/bid",
  "min_bid": 0.002
}
},
"content": {
  "categories": ["news", "opinion", "research"],
  "formats": ["text", "json", "html", "image"],
  "endpoints": [
    {
      "path": "/api/cpc/v1/content",
      "description": "Main content API",
      "methods": ["GET"],
      "parameters": {
        "id": "string",
        "format": ["json", "html"]
      }
    }
  ]
}
```

```
    }
  }
],
"restrictions": {
  "geo": ["US", "EU"],
  "bot_types": ["commercial", "academic"]
},
"logging": {
  "endpoint": "https://publisher.com/cpc/log",
  "required_fields": ["bot_id", "url", "timestamp", "size"],
  "signature_method": "HMAC-SHA256"
},
"terms": {
  "url": "https://publisher.com/cpc/terms",
  "last_updated": "2025-05-01"
}
}
```

Bot Processes Response:

- Checks `authentication.method` and requests an API key via `registration_url`.
- Evaluates `pricing.tiers` to determine if the crawl is cost-effective.
- Configures crawl to respect `rate_limits` and `restrictions`.

Bot Initiates Crawl:

bash

Unset

```
GET https://publisher.com/api/cpc/v1/content/123?api_key=XYZ
```

Publisher LLM Ingest API

The LLM Ingest API enables AI operators to query publisher content by submitting user prompts or queries. The API returns relevant content or paths to content (e.g., URLs) and supports billing based on pre-agreed pricing or dynamic bidding. All requests require a partner key, obtained through a prior business relationship with the publisher. This recommended API documentation provides a standardized interface for AI operators to interact with the LLM Ingest API, covering authentication, query submission, bidding, and billing.

Authentication

- **Method:** Partner Key (API Key)
- **Header:** X-Partner-Key: <your-partner-key>
- **Registration:** Obtain a partner key by registering at <https://publisher.com/llm-ingest/register>. A business relationship (e.g., contract) is required.
- **Scopes:**
 - `query_content`: Query and retrieve content.
 - `bid_content`: Submit bids for premium content.
- **Rate Limits:**
 - 100 requests/minute
 - 10,000 requests/day
- **Security:** All endpoints use HTTPS. Invalid or missing keys return a 401 Unauthorized response.

Endpoints

The API supports following endpoints:

Discovery Endpoint

Retrieve metadata about the API, including pricing, content types, and access rules.

- **URL:** `/info`
- **Method:** GET
- **Headers:**
 - X-Partner-Key: <your-partner-key> (optional for public discovery)

- **Response:**
 - **Status Codes:**
 - 200 OK: Successful response.
 - 401 Unauthorized: Invalid or missing partner key (if required).
 - **Body (application/json):**
 - json

JSON

```
{
  "api_version": "1.0",
  "publisher": {
    "name": "Example News",
    "id": "pub_12345",
    "contact": "llm-ingest@example.com"
  },
  "authentication": {
    "method": "partner_key",
    "registration_url": "https://publisher.com/llm-ingest/register"
  },
  "pricing": {
    "currency": "USD",
    "per_query_rate": 0.01,
    "content_rates": [
      {
        "content_type": "article",
        "rate": 0.05,
        "description": "Full-text news articles"
      },
      {
        "content_type": "report",
        "rate": 0.10,
        "description": "PDF research reports"
      }
    ]
  },
  "subscriptions": [
    {
      "plan_id": "pro",
      "name": "Pro Plan",
      "cost": 500,
    }
  ]
}
```

```

        "queries": 10000,
        "period": "monthly",
        "description": "Up to 10,000 queries/month"
    }
],
"dynamic_pricing": {
    "enabled": true,
    "bidding_endpoint": "/bid",
    "min_bid": 0.10
}
},
"content": {
    "categories": ["news", "reports"],
    "formats": ["json", "pdf", "html"],
    "query_endpoint": "/query"
},
"logging": {
    "endpoint": "/log",
    "required_fields": ["partner_key", "query_id", "timestamp",
"content_ids"]
},
"terms": {
    "url": "https://publisher.com/llm-ingest/terms",
    "last_updated": "2025-05-01"
}
}
}

```

Query Endpoint

Submit a user prompt or query to retrieve content or content paths.

- **URL:** /query
- **Method:** POST
- **Headers:**
 - X-Partner-Key: <your-partner-key> (required)
 - Content-Type: application/json
- **Request Body** (application/json):
- json

JSON

```
{
  "query": "string",
  "content_type": ["string"],
  "format": "string",
  "max_results": integer,
  "context": {
    "user_location": "string",
    "language": "string"
  }
}
```

- `query` (required): User prompt or search query (e.g., "Latest news on renewable energy").
- `content_type` (optional): Desired content types (e.g., ["article", "report"]).
- `format` (optional): Response format (e.g., "json", "pdf", "html"). Default: "json".
- `max_results` (optional): Maximum results to return (1–10). Default: 5.
- `context` (optional): Additional metadata (e.g., user location, language).
- **Response:**
 - **Status Codes:**
 - 200 OK: Query processed successfully.
 - 400 Bad Request: Invalid query or parameters.
 - 401 Unauthorized: Invalid or missing partner key.
 - 429 Too Many Requests: Rate limit exceeded.
 - **Body** (application/json):
 - json

JSON

```
{
  "status": "success",
  "query_id": "q_789",
  "results": [
    {
      "content_id": "art_123",
```

```

    "title": "Solar Energy Breakthroughs in 2025",
    "content": "string",
    "url": "https://publisher.com/content/art_123",
    "cost": 0.05,
    "content_type": "article",
    "format": "json"
  },
  {
    "content_id": "rep_456",
    "title": "Renewable Energy Report",
    "url": "https://publisher.com/s3/rep_456.pdf",
    "cost": 0.10,
    "content_type": "report",
    "format": "pdf"
  }
],
"total_cost": 0.15,
"billing_id": "bill_101"
}

```

- **content:** Full content (if small, e.g., article text). Omitted for large content.
- **url:** Secure, time-limited URL for large content (e.g., PDFs, datasets).
- **cost:** Cost per result (USD).
- **total_cost:** Sum of costs for all results.
- **billing_id:** Unique ID for billing purposes.

Bidding Endpoint

Submit a bid for premium or exclusive content.

- **URL:** /bid
- **Method:** POST
- **Headers:**
 - X-Partner-Key: <your-partner-key> (required)
 - Content-Type: application/json
- **Request Body** (application/json):

- json

JSON

```
{
  "content_id": "string",
  "bid_amount": number,
  "currency": "string",
  "expires": "string"
}
```

- content_id (required): ID of the desired content (e.g., "art_123").
- bid_amount (required): Bid amount (e.g., 0.07).
- currency (optional): Currency (e.g., "USD"). Default: "USD".
- expires (optional): Bid expiration time (ISO 8601, e.g., "2025-05-27T18:00:00Z").
- **Response:**
 - **Status Codes:**
 - 200 OK: Bid processed (accepted or rejected).
 - 400 Bad Request: Invalid content ID or bid.
 - 401 Unauthorized: Invalid or missing partner key.
 - **Body** (application/json):
 - json

JSON

```
{
  "status": "accepted",
  "content_id": "art_123",
  "final_cost": 0.07,
  "access_url": "https://publisher.com/content/art_123?token=abc",
  "expires": "2025-05-27T18:30:00Z"
}
```

or

- json

```
JSON
{
  "status": "rejected",
  "content_id": "art_123",
  "reason": "Bid below minimum ($0.10)"
}
```

Logging Endpoint

Submit query logs for billing verification (optional, per publisher requirements).

- **URL:** /log
- **Method:** POST
- **Headers:**
 - X-Partner-Key: <your-partner-key> (required)
 - Content-Type: application/json
- **Request Body** (application/json):
- json

```
JSON
{
  "partner_key": "string",
  "query_id": "string",
  "timestamp": "string",
  "content_ids": ["string"],
  "total_cost": number,
  "signature": "string"
}
```

- partner_key (required): Partner key for authentication.
- query_id (required): ID from query response.
- timestamp (required): ISO 8601 timestamp (e.g., "2025-05-27T16:51:00Z").

- `content_ids` (required): List of content IDs returned.
- `total_cost` (required): Total cost reported by client.
- `signature` (optional): HMAC-SHA256 signature for log integrity.
- **Response:**
 - **Status Codes:**
 - 200 OK: Log accepted.
 - 400 Bad Request: Invalid log data.
 - 401 Unauthorized: Invalid or missing partner key.
 - **Body** (application/json):
 - `json`

```
JSON
{
  "status": "accepted",
  "log_id": "log_456"
}
```

Pricing Options

Here are some suggested pricing options. But we expect the marketplace will evolve to determine how to price content.

- **Pre-Agreed Pricing:**
 - **Per-Query Rate:** \$0.01 per query.
 - **Rate per token:** \$0.001 per 1000 tokens.
 - **Content Rates:**
 - Article: \$0.05 per item.
 - Report: \$0.10 per item
 - **Subscriptions:**
 - Pro Plan: \$500/month for 10,000 queries.
 - Rates are set during partner registration via contract.
- **Dynamic Pricing/Bidding:**
 - Enabled for premium content (e.g., breaking news, proprietary datasets).
 - Minimum bid: \$0.10 (varies by content).
 - Use the `/bid` endpoint to submit bids.

Content Access

- **Categories:** News, Reports
- **Formats:** JSON, PDF, HTML
- **Restrictions:**
 - Geographic: US, EU only.
 - Bot Types: Commercial, Academic.
 - Usage: No archival or resale without explicit permission (per contract).
- **Delivery:**
 - Small content (e.g., articles) returned directly in response.
 - Large content (e.g., datasets) provided via secure, time-limited URLs.

Error Responses

- **Format** (application/json):
- `json`

JSON

```
{  
  "error": "string",  
  "code": integer,  
  "details": "string"  
}
```

- **Common Errors:**
 - 401 Unauthorized: { "error": "Invalid partner key", "code": 401 }
 - 429 Too Many Requests: { "error": "Rate limit exceeded", "code": 429 }
 - 400 Bad Request: { "error": "Invalid query", "code": 400 }

Example Requests

Query Content

bash

Shell

```
curl -X POST https://publisher.com/api/llm-ingest/v1/query \  
  -H "X-Partner-Key: XYZ" \  
  -H "Content-Type: application/json" \  
  -d '{  
    "query": "Latest news on renewable energy",  
    "content_type": ["article"],  
    "format": "json",  
    "max_results": 5  
  }'
```

Response:

json

JSON

```
{  
  "status": "success",  
  "query_id": "q_789",  
  "results": [  
    {  
      "content_id": "art_123",  
      "title": "Solar Energy Breakthroughs in 2025",  
      "content": "Full article text...",  
      "cost": 0.05,  
      "content_type": "article",  
      "format": "json"  
    }  
  ],  
  "total_cost": 0.05,  
  "billing_id": "bill_101"  
}
```

Submit Bid

bash

Shell

```
curl -X POST https://publisher.com/api/llm-ingest/v1/bid \  
  -H "X-Partner-Key: XYZ" \  
  -d '{  
    "bid": "bid_123",  
    "content_id": "art_123",  
    "cost": 0.05,  
    "content_type": "article",  
    "format": "json"  
  }'
```

```
-H "Content-Type: application/json" \  
-d '{  
  "content_id": "art_123",  
  "bid_amount": 0.07,  
  "currency": "USD",  
  "expires": "2025-05-27T18:00:00Z"  
}'
```

Response:

json

```
JSON  
{  
  "status": "accepted",  
  "content_id": "art_123",  
  "final_cost": 0.07,  
  "access_url": "https://publisher.com/content/art_123?token=abc",  
  "expires": "2025-05-27T18:30:00Z"  
}
```

Brand LLM Ingest API

Overview

The LLM Ingest API enables AI operators (e.g., LLMs, AI agents) to query a brand's content (e.g., product information, FAQs, or promotional materials) using natural language prompts. The API is similar to the publisher API but without the pricing component. The API supports conversational interactions, returning content or content paths (e.g., URLs) to enhance brand visibility and customer engagement.

Authentication

- **Method:** Partner Key (API Key)
- **Header:** `X-Partner-Key: <your-partner-key>`
- **Registration:** Obtain a partner key by registering at <https://examplebrand.com/llm-ingest/register>. A business relationship (e.g., contract or agreement) is required to ensure alignment with brand guidelines.
- **Scopes:**
 - `query_content`: Query and retrieve content.
- **Rate Limits:**
 - 100 requests/minute
 - 10,000 requests/day
- **Security:** All endpoints use HTTPS. Invalid or missing keys return a 401 `Unauthorized` response.

Endpoints

The API supports following endpoints:

Discovery Endpoint

Retrieve metadata about the API, including available content types, formats, and access rules.

- **URL:** `/info`
- **Method:** `GET`
- **Headers:**

- X-Partner-Key: <your-partner-key> (optional for public discovery)
- **Response:**
 - **Status Codes:**
 - 200 OK: Successful response.
 - 401 Unauthorized: Invalid or missing partner key (if required).
 - **Body** (application/json):
 - json

JSON

```
{  
  
  "api_version": "1.0",  
  
  "brand": {  
  
    "name": "ExampleBrand",  
  
    "id": "brand_exb_123",  
  
    "contact": "llm-ingest@examplebrand.com"  
  
  },  
  
  "authentication": {  
  
    "method": "partner_key",  
  
    "registration_url": "https://examplebrand.com/llm-ingest/register"  
  
  },  
  
  "content": {  
  
    "categories": ["products", "faqs", "promotions"],  
  
    "formats": ["json", "html", "pdf"],  
  
    "query_endpoint": "/query"  
  
  }  
}
```

```
},  
  
"logging": {  
  
  "endpoint": "/log",  
  
  "required_fields": ["partner_key", "query_id", "timestamp",  
"content_ids"]  
  
},  
  
"terms": {  
  
  "url": "https://examplebrand.com/llm-ingest/terms",  
  
  "last_updated": "2025-05-01"  
  
}  
  
}
```

Query Endpoint

Submit a user prompt or query to retrieve brand content or content paths.

- **URL:** /query
- **Method:** POST
- **Headers:**
 - X-Partner-Key: <your-partner-key> (required)
 - Content-Type: application/json
- **Request Body** (application/json):
- json

JSON

```
{  
  
  "query": "string",
```

```
"content_type": ["string"],

"format": "string",

"max_results": integer,

"context": {

  "user_location": "string",

  "language": "string"

}

}
```

- `query` (required): User prompt or search query (e.g., "Tell me about ExampleBrand's latest product").
- `content_type` (optional): Desired content types (e.g., ["products", "faqs"]). Default: all types.
- `format` (optional): Response format (e.g., "json", "html", "pdf"). Default: "json".
- `max_results` (optional): Maximum results to return (1–10). Default: 5.
- `context` (optional): Additional metadata (e.g., user location, language) for personalized responses.
- **Response:**
 - **Status Codes:**
 - 200 OK: Query processed successfully.
 - 400 Bad Request: Invalid query or parameters.
 - 401 Unauthorized: Invalid or missing partner key.
 - 429 Too Many Requests: Rate limit exceeded.
 - **Body** (application/json):
 - json

```
JSON
{
```

```
"status": "success",

"query_id": "q_789",

"results": [

  {

    "content_id": "prod_123",

    "title": "ExampleBrand Product X Specifications",

    "content": "Product X features advanced technology...",

    "url": "https://examplebrand.com/content/prod_123",

    "content_type": "product",

    "format": "json"

  },

  {

    "content_id": "faq_456",

    "title": "How to Contact ExampleBrand Support",

    "url": "https://examplebrand.com/s3/faq_456.pdf",

    "content_type": "faq",

    "format": "pdf"

  }

]

}
```

- `content`: Full content for small items (e.g., product descriptions). Omitted for large content.
- `url`: Secure, time-limited URL for large content (e.g., PDFs, brochures).
- `query_id`: Unique ID for tracking the query.

Logging Endpoint

Submit query logs for analytics and verification (optional, per brand requirements).

- **URL:** `/log`
- **Method:** POST
- **Headers:**
 - `X-Partner-Key`: `<your-partner-key>` (required)
 - `Content-Type`: `application/json`
- **Request Body** (application/json):
 - `json`

JSON

```
{
  "partner_key": "string",
  "query_id": "string",
  "timestamp": "string",
  "content_ids": ["string"]
}
```

- `partner_key` (required): Partner key for authentication.
- `query_id` (required): ID from query response.
- `timestamp` (required): ISO 8601 timestamp (e.g., "2025-05-27T15:29:00Z").
- `content_ids` (required): List of content IDs returned.
- **Response:**
 - **Status Codes:**
 - 200 OK: Log accepted.
 - 400 Bad Request: Invalid log data.

- 401 Unauthorized: Invalid or missing partner key.
- **Body** (application/json):
- json

JSON

```
{  
  
  "status": "accepted",  
  
  "log_id": "log_456"  
  
}
```

Content Access

- **Categories:**
 - `products`: Product details (e.g., specs, features).
 - `faqs`: Frequently asked questions and answers.
 - `promotions`: Marketing materials and offers.
- **Formats:** JSON, HTML, PDF
- **Restrictions:**
 - Geographic: Global access unless specified in contract.
 - Usage: No archival, resale, or modification of content without permission (per terms).
- **Delivery:**
 - Small content (e.g., product specs) returned directly in response.
 - Large content (e.g., brochures) provided via secure, time-limited URLs.

Error Responses

- **Format** (application/json):
- json

JSON

```
{
```

```
"error": "string",  
  
"code": integer,  
  
"details": "string"  
  
}
```

- **Common Errors:**

- 401 Unauthorized: { "error": "Invalid partner key", "code": 401 }
- 429 Too Many Requests: { "error": "Rate limit exceeded", "code": 429 }
- 400 Bad Request: { "error": "Invalid query", "code": 400 }

Example Requests

Query Content

bash

```
Shell  
  
curl -X POST https://examplebrand.com/api/llm-ingest/v1/query \  
  
-H "X-Partner-Key: XYZ" \  
  
-H "Content-Type: application/json" \  
  
-d '{  
  
  "query": "Tell me about ExampleBrand's latest product",  
  
  "content_type": ["product"],  
  
  "format": "json",  
  
  "max_results": 5
```

```
}'
```

Response:

json

JSON

```
{
  "status": "success",
  "query_id": "q_789",
  "results": [
    {
      "content_id": "prod_123",
      "title": "ExampleBrand Product X Specifications",
      "content": "Product X features advanced technology...",
      "content_type": "product",
      "format": "json"
    }
  ]
}
```

Submit Log

bash

Shell

```
curl -X POST https://examplebrand.com/api/llm-ingest/v1/log \
  -H "X-Partner-Key: XYZ" \
```

```
-H "Content-Type: application/json" \  
  
-d '{  
  
  "partner_key": "XYZ",  
  
  "query_id": "q_789",  
  
  "timestamp": "2025-05-27T15:29:00Z",  
  
  "content_ids": ["prod_123"]  
  
}'
```

Response:

json

```
JSON  
  
{  
  
  "status": "accepted",  
  
  "log_id": "log_456"  
  
}
```

NLWeb option for publishers and brands

Microsoft recently announced the NLWeb(github.com/microsoft/NLWeb) initiative to build conversational interfaces for websites. NLweb allows website owners to leverage existing work like schema.org and RSS infrastructure to create a natural language interface. In addition, it uses MCP (Model Context Protocol) that can be used for both human and machine interface. With NLWeb, website content owners can setup interfaces for humans as well as AI agents and LLMs.

Here we present this as an option for brands and publishers to set up a LLM and AI agent friendly system to manage and control how their content is used.

NLWeb vs LLM Ingest API

Below is a quick comparison of NLWeb vs LLM Ingest APIs.

Feature	NLWeb	LLM Ingest API (ExampleBrand)
Purpose	Enables conversational web interfaces for users and AI agents via MCP.	Enables AI agents to query brand content with a focus on secure, free access.
Query Mechanism	“Ask” method for natural language queries, returning JSON based on Schema.org.	/query endpoint for natural language prompts, returning content or URLs.
Authentication	No built-in authentication; relies on website-level security (e.g., API keys).	Partner key system (Partner-Key) for secure access, requiring a business relationship.
Pricing/Billing	No pricing or billing mechanism; content is typically free.	No pricing or bidding, designed for free content delivery to trusted partners.
Logging	No explicit logging endpoint; analytics depend on implementation.	/log endpoint for analytics, tracking queries and content usage.
Content Control	Publishers choose content via Schema.org or custom configs.	Brands specify content types (e.g., products, FAQs) and restrictions via contract.

Scalability	Cross-platform (Windows, macOS, Linux), cloud-to-laptop support.	Cloud-based with CDN support for high query volumes.
Integration	MCP compatibility for AI ecosystems; aligns with NLWeb adopters (e.g., Shopify).	Custom API tailored for ExampleBrand, integrable with NLWeb via MCP.

Advantages of NLWeb

- **Standardized Protocol:**
 - NLWeb’s MCP and Schema.org integration provide a standardized way to expose content, ensuring compatibility with major LLMs (e.g., OpenAI, Anthropic, xAI). This reduces development overhead compared to building a custom API
- **Ecosystem Support:**
 - Backed by Microsoft and early adopters (e.g., O’Reilly, TripAdvisor), NLWeb offers a growing ecosystem, increasing the likelihood of AI agents adopting it for querying brand content.
- **No Crawling Required:**
 - Unlike traditional web crawling, NLWeb uses structured data (e.g., Schema.org), allowing publishers and brands to provide precise content (e.g., product specs) to LLMs.
- **Open-Source Flexibility:**
 - Available on GitHub (github.com/microsoft/NLWeb, 3.4k stars as of May 26, 2025), NLWeb allows brands to customize implementations, such as adding authentication or analytics, to include API partner key and logging features.
- **Enhanced Visibility:**
 - By integrating with AI assistants via MCP, publishers and brands can reach users through conversational platforms, boosting engagement without direct monetization.

Limitations of NLWeb

- **Lack of Built-In Authentication:**
 - NLWeb does not provide a native partner key system, unlike LLM Ingest API’s `X-Partner-Key` header. Brands must implement their own authentication (e.g., API keys, OAuth) to restrict access to trusted AI operators, which adds complexity.

- **No Logging or Analytics:**
 - NLWeb lacks a dedicated logging endpoint like `/log` endpoint, requiring brands to build custom analytics to track query usage or optimize content, which is critical for understanding customer interests.
- **No Monetization Framework:**
 - NLWeb's lack of a pricing or bidding framework makes it less adaptable for publishers to monetize content. But it works well for brands if they want to distribute their content free of cost
- **Implementation Effort:**
 - Although NLWeb provides reference implementations, brands with complex content (e.g., dynamic product catalogs) may need significant setup to structure data with Schema.org, whereas LLM Ingest API allows for simpler, brand-specific content categories.
- **Early Stage Adoption:**
 - As a new initiative, NLWeb's ecosystem is still growing, with potential risks of limited AI agent adoption compared to your LLM ingest API, which can be tailored to specific partners

Recommendations for using NLWeb option

- **Adopt NLWeb as a Foundation:**
 - Use NLWeb's MCP server and Schema.org integration to expose structured content (e.g., product details, FAQs) to LLMs, leveraging its compatibility with AI ecosystems.
 - Example: Structure product data with Schema.org's `Product` type and implement NLWeb's "ask" method to handle queries like "What are ExampleBrand's eco-friendly products?"
- **Integrate with LLM Ingest API:**
 - Extend NLWeb with your `/query` and `/log` endpoints to add authentication (`X-aposent: -Partner-Key`) and analytics, ensuring secure and trackable access.
 - Example: Map NLWeb's "ask" method to your `/query` endpoint, returning JSON responses with content or URLs.
- **Establish Business Relationships:**
 - Require AI operators to register for a partner key via `https://example.com/llm-ingest/register`, aligning with your API's trust model and ensuring compliance with brand guidelines.
 - Example: Partner with AI providers to ensure their agents (e.g., OpenAI, Grok, Copilot) query content accurately.
- **Leverage Free Access for Visibility:**

- Offer restricted content for free, as in your API design, to maximize reach through AI assistants, driving traffic to website or products.
- Example: Respond to queries like “Brand promotions” with links to marketing campaigns, increasing customer engagement.
- **Use Analytics to Optimize:**
 - Implement `/log` endpoint to track query trends (e.g., popular products) and refine content offerings, compensating for NLWeb’s lack of built-in analytics.
 - Example: Log queries to identify demand for specific FAQs and update content accordingly.

Example Integration

- **NLWeb Setup:** ExampleBrand deploys an NLWeb MCP server, structuring product data with Schema.org (e.g., `Product` with attributes like `name`, `description`).
- **LLM Ingest API Overlay:**
 - **Discovery:** Expose NLWeb metadata via `/info`, adding authentication details (`partner_key` requirement).
 - **Query:** Map NLWeb’s “ask” method to `/query`, requiring a partner key and returning JSON with content or URLs.
 - **Logging:** Use `/log` to record queries for analytics, e.g., `{ "query_id": "q_789", "content_ids": ["prod_123"] }`.
- **Sample Query:**
- `bash`

Shell

```
curl -X POST https://examplebrand.com/api/llm-ingest/v1/query \

-H "X-Partner-Key: XYZ" \

-H "Content-Type: application/json" \

-d '{"query": "Latest ExampleBrand products", "content_type": ["product"], "format": "json"}'
```

Response:

- json

JSON

```
{  
  
  "status": "success",  
  
  "query_id": "q_789",  
  
  "results": [  
  
    {  
  
      "content_id": "prod_123",  
  
      "title": "ExampleBrand Product X",  
  
      "content": "Advanced technology product...",  
  
      "content_type": "product",  
  
      "format": "json"  
  
    }  
  
  ]  
  
}
```

Conclusion

NLWeb is a strong option for publishers and brands alike to provide content to LLMs, offering a standardized, conversational interface that aligns with LLM Ingest API's goals of query-driven content delivery. Its MCP integration and open-source nature make it accessible and scalable, but it lacks native authentication and analytics. By combining NLWeb's protocol with LLM Ingest API's endpoints, website owners can securely and effectively engage AI ecosystems, boosting visibility and customer interaction.

Conclusion

The integration of Large Language Models (LLMs) and AI agents presents significant opportunities for publishers and brands alike. The traditional web model is disrupted by AI-driven search summaries and increased bot scraping, leading to traffic and revenue declines for publishers. Brands face risks to their reputation and messaging if AI-generated content is not properly managed.

However, this evolution also provides avenues for adaptation and growth.

Publishers can implement mechanisms to control content access, provide LLM-friendly discovery through methods like `llms.txt`, and monetize content through Cost per Crawl (CPCr) or LLM Ingest APIs.

Brands can ensure accurate and brand-aligned information by understanding and managing content access, providing LLM-friendly discovery, and leveraging the LLM Ingest API. Technologies like NLWeb further enable conversational interfaces for controlled content delivery.

In summary, while the rise of AI poses threats to existing models, proactive measures such as implementing content access controls, optimizing for LLM discovery, and developing structured APIs can ensure the sustainability and relevance of publishers and brands in the AI-driven future.

We are calling on the industry to embrace these changes and adopt innovative approaches.

Content creators and businesses can effectively navigate the evolving digital ecosystem and capitalize on new opportunities for engagement and monetization by leaning in and standardizing interoperable methods to further develop:

- Bots and agents access control standardization
- Content discovery: content meta data declarations and `llms.txt`
- Cost per Crawl (CPCr) monetization API
- LLMs Ingest APIs
- NLweb integration

We look forward to industry feedback and organizing a workshop to determine next steps in the evolution of the LLM integration framework.