Natural Language Interaction Protocol:

A Universal Application-Level Protocol for the Age of Generative AI

(ECMA-TC56, AI Alliance Affiliated Project)

Ranjan Sinha
IBM Research

IBM

Cisco

Red Hat

Hitachi

ServiceNow

Fordham University

University of Michigan

Purdue University

University of Delaware

University at Buffalo

Pennsylvania State University

Indiana University

University of Illinois Urbana-Champaign

SRI International

Collaborators

IBM

- Dinesh Verma
- Jonathan Lenchner
- Abhay Ratnaparkhi
- Ranjan Sinha

RedHat

- Eric Erlandson
- Sanjay Aiyagari

Cisco

- Ashish Kundu

Hitachi

- Yohei Kawaguchi

ServiceNow

- Sean Hughes

SRI International

- Yan-Ming Chiou

Independents

- Raj Doodhiawala
- Tom Sheffler

University at Buffalo

- Jinjun Xiong

University of Delaware

- Chien-Chung Shen
- Mathews Mauriello

University of Michigan

- Sugih Jamin

Purdue University

- Elisa Bertino

Indiana University / University of Illinois Urbana-Champaign

- Luyi Xing

Fordham University

- Mahomed Rahouti

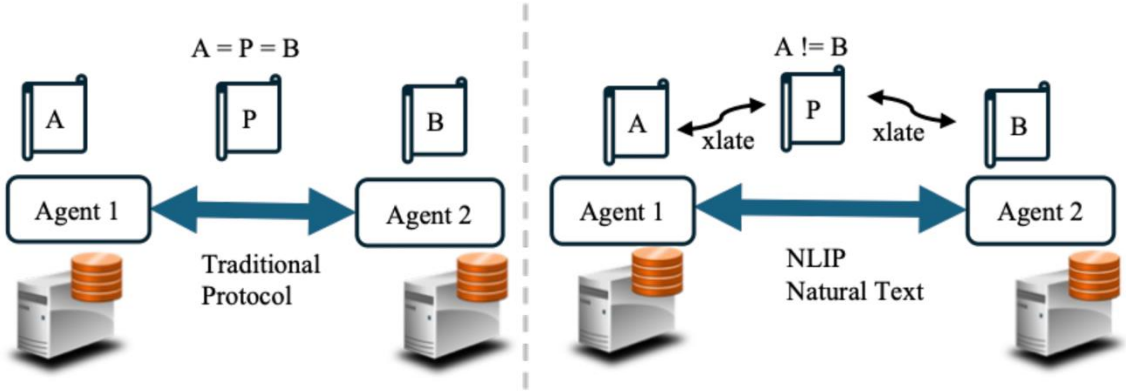
Pennsylvania State University

- Winpeng Yin

Motivation

- Several application-level protocols and APIs with rigid schemas in use.
- Challenges with integration, interoperability, and version management.

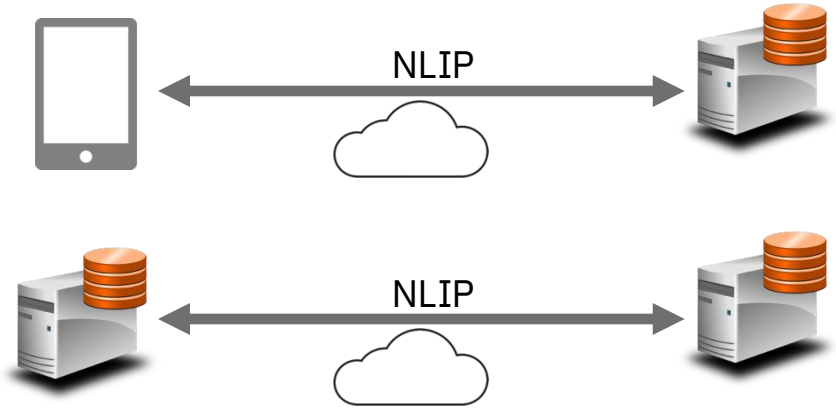
Could we apply generative AI to design smarter, more adaptable protocols?



Vision

NLIP provides a simple, open, standard, common protocol for apps, agents, and services to communicate with each other and to us.

- Assumes GenAI capability on both end-points.
- Follows request-response paradigm, to ensure that the meaning is refined and understood.



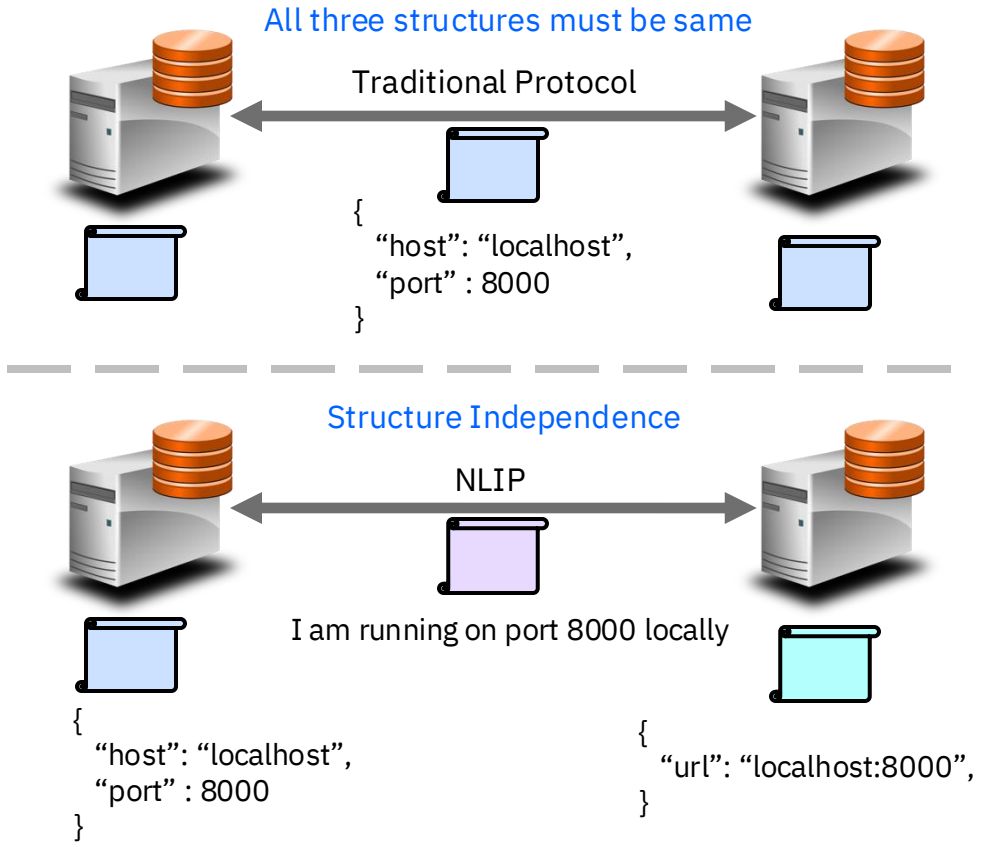
Requirements for Protocol

- Should be secure
- Should enable various safeguards such as data privacy, data usage policy, and DDoS prevention
- Should be implementable over various underlying transports and API styles
- Should support multimedia content exchange
- Should permit communication efficiency
- Should enable implementation in various languages
- Should be standardized in an open forum

NLIP uses Generative AI for structure independent communications

- Application-level protocols define the structure
  - Forces a rigid API
  - Version upgrade issues
  - Limits extensibility
  - Tight coupling between systems

- NLIP allows structure independence
  - Information exchanged on the wire will be translated to a structure local to each communicating party
  - GenAI is good at parsing unstructured text to structured text



NLIP Scope

In-Scope

- Communication between two AI-Agents
  - Both across organizations and within organization
- Communication between a human and an AI-Agent
  - Both across organizations and within organization

Out of Scope

- Communication between legacy software
  - Both across organizations and within organization
- Communication between a legacy software (not using AI) client and an AI-Agent
  - Both across organizations and within organization

NLIP is for AI-to-AI and human-to-AI conversations — not for old, pre-AI systems.

NLIP and other Protocols

There are multiple single vendor protocols proposed for different aspects of AI agent tasks such as resource management, tool calls, and discovery

– Anthropic MCP, IBM ACP, Google A2A, Cisco AGNTCY, …

NLIP can interoperate with them in two ways

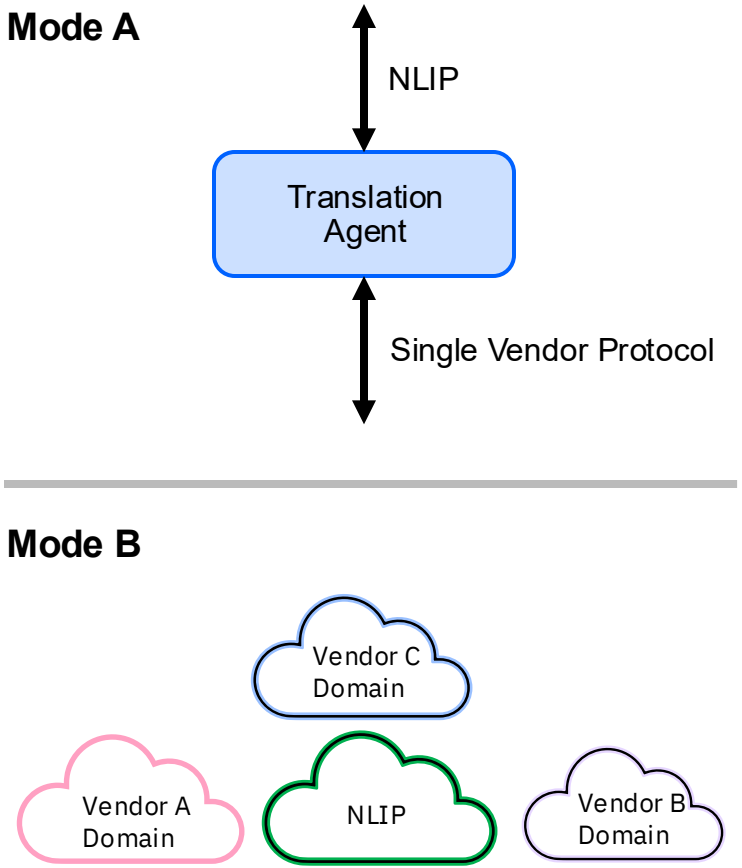
– Mode A:

  • NLIP provides Northbound API, vendor protocol provides Southbound API. Using a LLM to support translation.

– Mode B:

  • NLIP provides interoperability across different domains of single vendor protocols. Each domain may have multiple agents using different frameworks.

Eventually, the industry should converge to an open standard protocol such as NLIP



History Repeats Itself

Observation

The IT landscape today is similar to 1990s

1990s:

- Every business app had its own protocol, custom client, and bespoke integration.
- Pre-HTTP protocols: FTP, Telnet, Gopher, NNTP, client-server protocols for CRM, ERP …
- HTTP emerged as a simple, uniform, extensible protocol.
- HTTP unified Internet experience under a single, general-purpose interface – the web browser.
- Simplified application delivery, interoperability, unleashed Internet applications.

Today

- Fragmented ecosystem of AI agents, tools, APIs, and proprietary integrations.
- LLMs and AI agents don't all talk the same language.
- AI agents and chat are becoming a common interface for business solutions.
- If we can define a single standard protocol – HTTP for intelligent systems.
- Unleash scale and collaboration for Agentic AI applications.

NLIP will be for Agentic AI applications what HTTP was for the Internet.

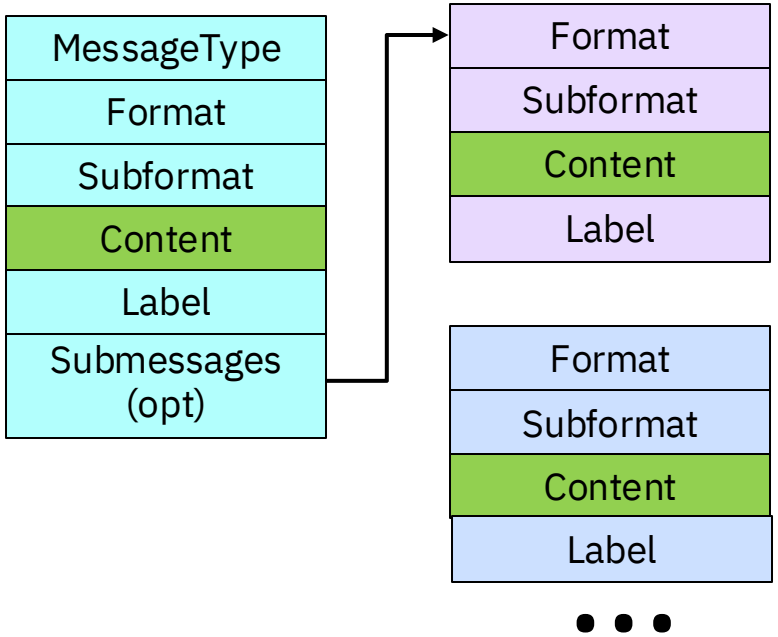
NLIP Schema – Simple and Concise

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "title": "NLIP Message Schema",
  "type": "object",
  "properties": {
    "MessageType": {
      "type": "string"
    },
    "Format": {
      "type": "string",
      "enum": ["text", "token", "structured", "binary", "location", "error", "generic"]
    },
    "Subformat": {
      "type": "string"
    },
    "Content": {
      "type": ["string", "number", "object", "array", "boolean", "null"]
    },
    "Submessages": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "Label": { "type": "string" },
          "Format": {
            "type": "string",
            "enum": ["text", "token", "structured", "binary", "location", "error", "generic"]
          },
          "Subformat": { "type": "string" },
          "Content": {
            "type": ["string", "number", "object", "array", "boolean", "null"]
          }
        },
        "required": ["Format", "Subformat", "Content"]
      }
    }
  },
  "required": ["Format", "Subformat", "Content"]
}
```

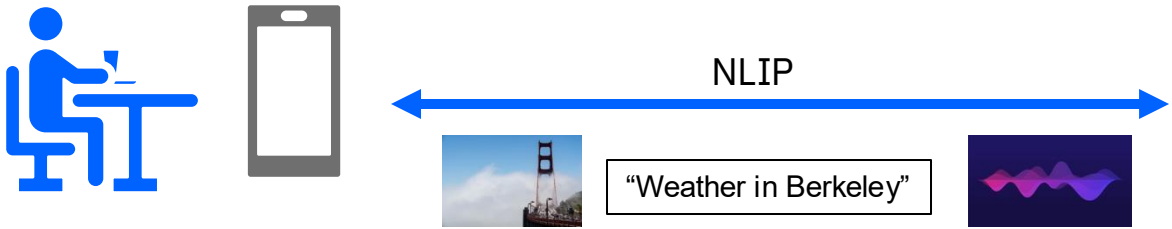
https://github.com/nlip-project/documents/blob/main/specification/nlip.json

NLIP Message Structure

- Each NLIP message has a Type (control/data/general)
- The basic structure of an NLIP Exchange is a Sub-Message
  - Format: what kind of content? (Text/JSON/Image/Video).
  - Subformat: More detail on the format. (Language for text, encoding for images).
  - Content: The actual message (“What is the weather in Berkeley tomorrow?”)
  - Label: simple name or tag (weather-info).
- NLIP messages can consist of a sequence of sub-messages



NLIP Message Format is Community-Driven



- Text
- Media
  - Audio
  - Video
  - Images
- Documents
- Structured Information

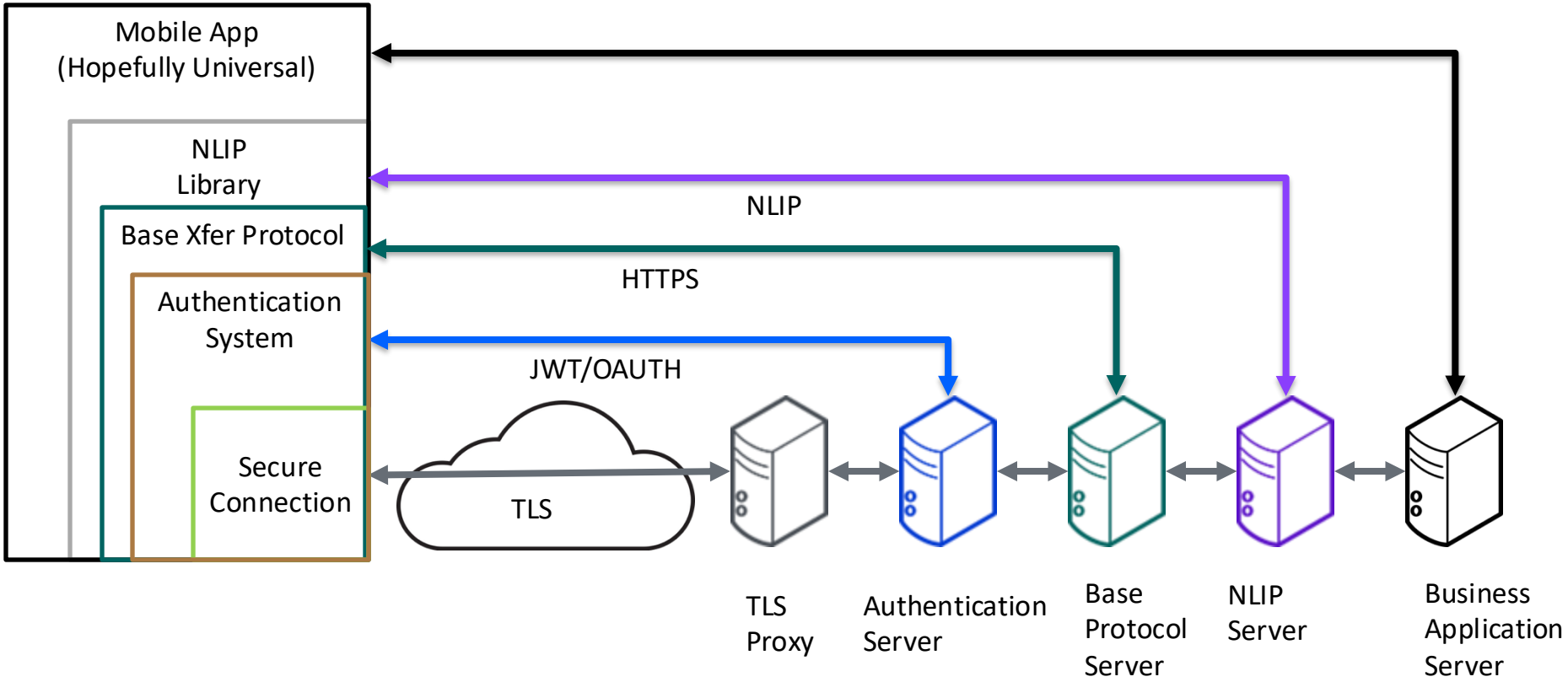
NLIP aims to solve the “Tower of Babel” problem in AI agent communication.

1. Agntcy: https://spec.acp.agntcy.org/#model/message
2. ACP: https://agentcommunicationprotocol.dev/core-concepts/message-structure
3. A2A: https://a2aproject.github.io/A2A/latest/specification/#64-message-object
4. Langchain: https://python.langchain.com/api_reference/core/messages/langchain_core.messages.ai.AIMessage.html
5. Autogen: https://github.com/microsoft/autogen/blob/main/python/packages/autogen-agentchat/src/autogen_agentchat/messages.py

NLIP Salient Features

- Primary mode of interaction is natural language.
  - Could be typed text or spoken text with speech to text conversion.
  - Promotes an intuitive human-centric communication.
- Vision, Audio and other modalities are supported.
- Structured text supported but not intended as a primary interaction mode.
- No custom configuration – All configuration and policy exchanges made using natural language.
- The AI models used by different end-points (clients and servers) in a NLIP session can be different.

NLIP Reference Architecture



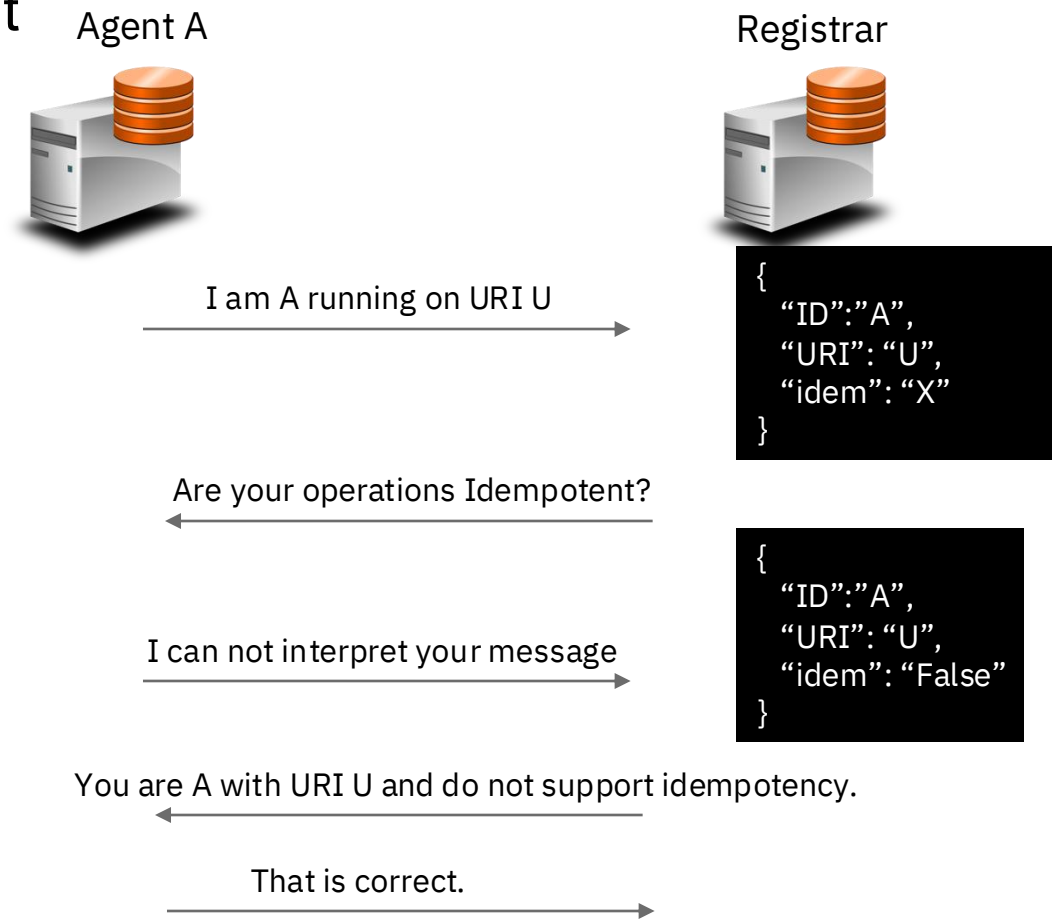
NLIP integrates seamlessly with existing system architectures.

Example: Version Management

NLIP can be used to interoperate among services that have different versions of internal representation.

Example:

- Service R is a registration service:
  - Each registered agent has fields id, URI and Idempotency
- Agent A does not support idempotency field
  - Initial registration does not contain information about it



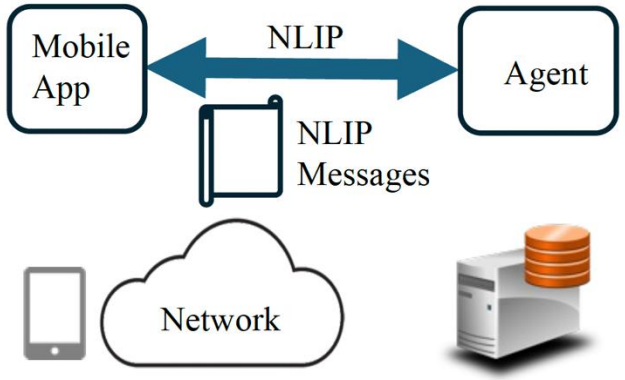
Use Case I: Chat Interaction

Chat client enquiring about the Ecma organization.


```
{
"format":"text",
"subformat":"english",
"content":"What is Ecma?”
}
```


Response from the chat Server.


```
{
"format":"text",
"subformat":"english",
"content":"Ecma International (Ecma) is an independent,
non-profit, global standards organization. It develops and
publishes international standards for information technology,
including specifications for programming languages, data
formats, and software licensing.”
}
```

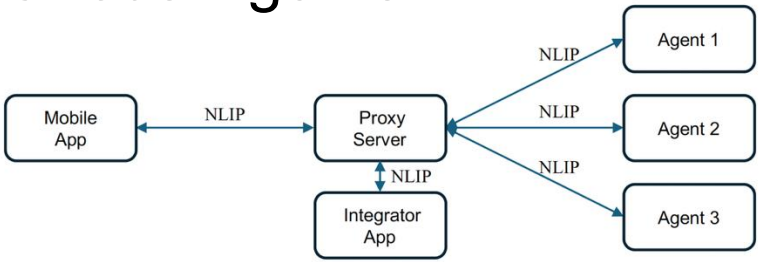



https://github.com/nlip-project/ecma_draft/blob/main/tc56-2025-018.pdf

Use Case II: Federation among Various Agents

1. Client sends a NLIP message to the proxy server

```
{
"format":"text",
"subformat":"english",
"content":"What is Ecma?"
}
```



2. Same message is sent out to three agents by the proxy server

Agent 1

```
{
"format":"text",
"subformat":"english",
"content":"Ecma International (Ecma) is an independent, non-profit, global standards organization."
}
```

Agent 2

```
{
"format":"text",
"subformat":"english",
"content":"ECMA (European Computer Manufacturers Association) International is a non-profit organization."
}
```

Agent 3

```
{
"format":"text",
"subformat":"english",
"content":"ECMA, or European Computer Manufacturers Association, is an organization."
}
```

3. Proxy server combines the responses from all the agents. It may send a request to an integrator agent using a NLIP messages with multiple submessages.

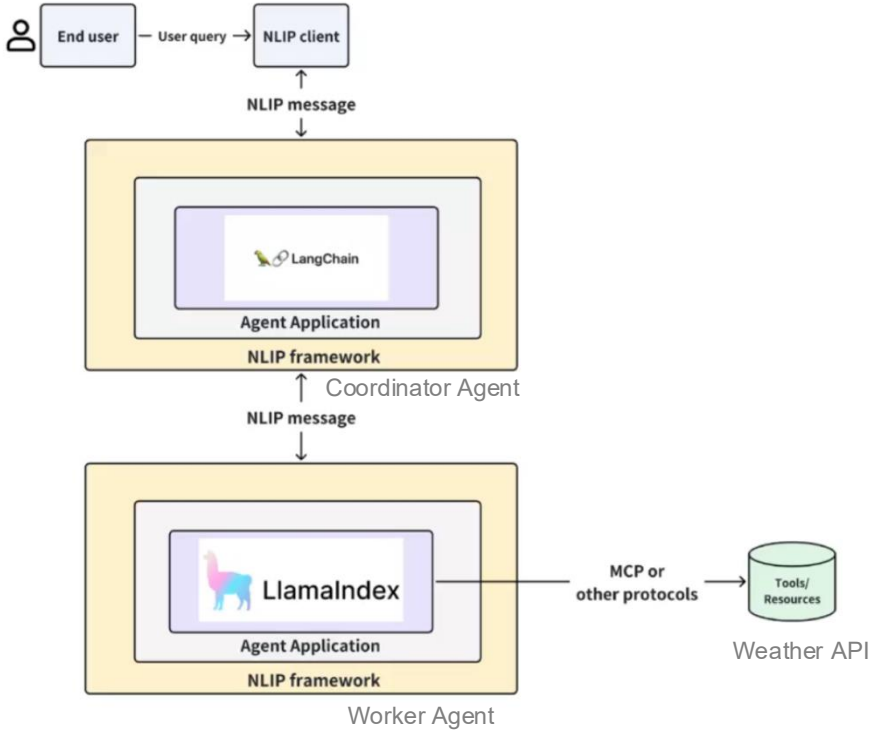
```
{
"messagetype":"Request",
"format":"text",
"subformat":"english",
"content":"Please combine the requests in the submessages",
"submessages":[
{
"format":"text",
"subformat":"English",
"content": "Ecma International (Ecma) is an independent, non-profit, global standards organization.
},
{
"format":"text",
"subformat":"English",
"content":"ECMA (European Computer Manufacturers Association) is a non-profit organization."
},
{"format":"text",
"subformat":"English",
"content":"ECMA, or European Computer Manufacturers Association, is an organization."
}
]}
```

Use Case III: Integrating with Agent Development Frameworks

- Works with popular Agentic AI frameworks like Langchain and LlamaIndex.
- Supports tool calling through standard protocols.
- Enables agent-to-agent communication.

Weather chatbot

- Coordinator agent receives a user query and delegates tasks to a worker agent, which accesses an external weather API.
- The agents use NLIP messages for their interactions.
- Coordinator aggregate responses from the worker and returns a synthesized answer to the user.



Use Case IV: Speech-to-Text for Customer Support

Customers get contextually relevant Reddit content just by speaking their question.

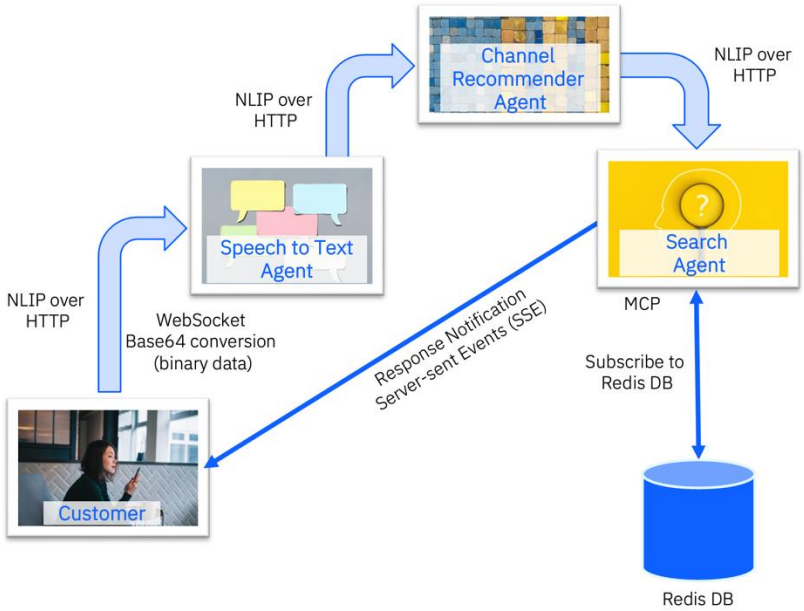
Voice Input: Audio is captured and sent via WebSocket as Base64-encoded data or as binary audio data using the NLIP message format.

Speech Processing: Speech-to-Text agent uses NVIDIA ASR model and forwards the text using NLIP protocol.

Smart Channel Selection: The Channel Recommender agent analyzes the query and identifies the most relevant Reddit channels to search.

Intelligent Search: Search Agent calls the Reddit APIs, pulls the most recent and relevant posts from each recommended channel, and caches results in Redis for real-time updates.

Seamless Integration: We use MCP (Model Context Protocol) to connect with Reddit's APIs, ensuring we get fresh, relevant content back to the user.



The entire flow uses NLIP messaging over HTTP, with Redis handling pub-sub for live updates.

NLIP Key Takeaways

Feature	Summary
Structure-Agnostic	NLIP is not tied to any predefined schema, allowing flexibility across diverse data formats.
Natural Language-Centric	Designed for expressing and interpreting intent in natural language, ideal for LLM-driven interactions.
Envelope Protocol	Acts as a wrapper that can carry other protocol-specific messages, enabling cross-protocol communication.
Built for Generative AI	Supports use of LLMs to translate or mediate between different agent message formats.
Design Principles	Simple, concise, open, and standardized — aiming to be a common protocol across agent ecosystems.

NLIP Status

- NLIP is being defined and standardized under ECMA TC-56.
- Draft specifications and initial implementations are available at: https://github.com/nlip-project.
- Comments and feedback are being solicited for revisions at: https://nlip-project.org.
- Formal standardization in the second half of 2025.

For more information or to get involved, please visit NLIP GitHub and contact us at contact@nlip-project.org.

THANK YOU

NLIP Security Guidelines and Best Practices

- Purpose: Provide a pragmatic, auditable checklist for securing NLIP-based multi-agent systems.
- Scope: Identity, transport, runtime behaviour, data storage, observability, governance, and incident response. Expanded to include regulatory-compliance mappings (e.g., EU AI Act), ethical considerations, and supply-chain enhancements based on 2025 best practices.
- Limitations: Excludes foundation-model internals, physical-datacentre safeguards, and national export controls.

https://github.com/nlip-project/security_guidelines

Support for Binary Data

- Direct delivery - For things like live audio, send the raw data straight through without any extra packaging. It's fast and efficient.
- Standard packaging - Wrap the file in a special format (Base64) with a shipping label that tells you what's inside - whether it's a video, image, or audio file.
- Smart shipping - Automatically choose the fastest delivery method for big files like videos. If that doesn't work, we fall back to the standard packaging method.

https://github.com/nlip-project/ecma_draft/blob/main/tc56-2025-018.pdf

Current Set of NLIP Messages

Format	Subformat	Notes
text	Language of text (e.g. English, French etc.)	Provides a hint to the other side, can be used for model selection
token	An opaque string – determined by token creator	Content is an opaque string – used by end-point to share information such as correlators or authentication data
structured	JSON, URI, XML	Content is structured format. Allows backward compatibility for display of graphical GUI
binary	Encoding – image, video, sensor etc.	Subformat is content-type/encoding
location	Text or GPS	Coordinate if GPS, description if text
generic	Open	Provided for future extensions