ecma INTERNATIONAL

Ecma/TC56/2025/020

2nd Draft Standard ECMA-XXX

1st Edition / July 2025

Binding of the Natural Language Interaction Protocol (NLIP) over WebSocket

Standard

Rue du Rhône 114 CH-1204 Geneva T: +41 22 849 6000 F: +41 22 849 6001

is the registered trademark of Ecma International

COPYRIGHT PROTECTED DOCUMENT

Contents

Page

Introduction

The technology of Generative AI (GAI) has the potential to be truly transformative to society. Despite some limitations such as “hallucinations,” the technology is capable of many functions, including but not limited to answering questions, translating, describing and summarizing multi-modal content, generating new content, and summarizing large volumes of information. This enables the creation of intelligent agents that can use AI to analyze data and provide new services.

A much bigger boost to the social benefits of generative AI technology can be obtained by interaction among different intelligent agents, which may be under the control of different organizations and users. The interaction among intelligent agents can unlock new economic and social value, just like the interactions among various Internet-based services was enabled with the advent of the web browser.

For the intelligent agents to interact with each other, there is a need for a standard common protocol that is used widely among interacting agents. This Standard specifies such a protocol which would ensure interoperability among various services that use AI based technology.

ECMA-XXX defines a Natural Language Interaction Protocol (NLIP).

This Standard describes the binding of NLIP protocol to a base transfer protocol which is using WebSocket.

This Ecma Standard was developed by Technical Committee 56 and was adopted by the General Assembly of <month> <year>.

Binding of the Natural Language Interaction Protocol over WebSocket

1 Scope

This specification defines how the Natural Language Interaction Protocol (NLIP) shall be implemented over the WebSocket protocol using CBOR (Concise Binary Object Representation, RFC 8949) for compact and efficient multimodal communication. It also describes a fallback to UTF-8 encoded JSON text frames for compatibility.

2 Conformance

A conformant implementation MUST:

- Support full NLIP message schema as defined in the NLIP JSON Schema.
- Encode/decode messages in CBOR format over binary WebSocket frames.
- Optionally fall back to UTF-8 JSON text frames for non-CBOR-capable peers.
- Support transmission of multimodal submessages, including raw binary content (e.g., audio, image).

3 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

IETF RFC 6455, *WebSocket Protocol*
[https://datatracker.ietf.org/doc/html/rfc6455]

IETF RFC 7049, *CBOR Data Model*
[https://datatracker.ietf.org/doc/html/rfc7049]

IETF RFC 7230, *Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing*
[https://datatracker.ietf.org/doc/rfc7230/]

IETF RFC 7240, *Prefer Header for HTTP*
[https://datatracker.ietf.org/doc/rfc7240/]

IETF RFC 8949, *Concise Binary Object Representation (CBOR)*
[https://datatracker.ietf.org/doc/html/rfc8949]

NLIP JSON Schema (ECMA-tc56)

4 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

4.1
NLIP
NLIP or Natural Language Interaction Protocol is the protocol defined in ECMA-XXX.

4.2
base transfer protocol
a transfer protocol is a communication protocol between two computer systems which supports an encrypted and authenticated transfer of data across those computer systems.

4.3
CBOR
Binary serialization format for structured data.

5 NLIP WebSocket endpoint

The server MUST expose a WebSocket endpoint at:

wss://<host>:<port>/nlip/ws

A conformant implementation of NLIP over HTTPS/REST will have the Server end-point running on a TCP Server Port. This port would be accessible using a URL defined as https://<server_name>:port/nlip.

5.1 Optional fallback endpoint

wss://<host>:<port>/nlip/ws/text

6 Message transmission

6.1 CBOR format

- Each WebSocket binary frame MUST contain a single NLIP message encoded in CBOR.
- CBOR Content fields MAY include:
  - String
  - Byte string (raw binary)
  - Array, Object (Map)
- Submessages are embedded in CBOR using the same schema.

6.2 Text fallback format

- If CBOR is not supported:
  - Use UTF-8 encoded JSON in WebSocket text frames.
  - Binary data MUST be base64-encoded.

7 Examples

Example 1: text + audio (CBOR)

NLIP message (in Python pseudo-code before CBOR encoding)

```
{
  "MessageType": "Request",
  "Format": "structured",
  "Subformat": "application/json",
  "Content": {"intent": "weather_query"},
  "Submessages": [
    {
      "Label": "transcription",
      "Format": "text",
      "Subformat": "en-US",
      "Content": "What's the weather in Austin tomorrow?"
    },
    {
      "Label": "audio",
      "Format": "binary",
      "Subformat": "audio/wav",
      "Content": b'\x52\x49\x46\x46...'  # Raw binary WAV
    }
  ]
}
```

This is encoded as a **single CBOR binary frame**. The audio submessage uses a byte string directly and no base64 encoding.

Example 2: image processing request (CBOR)

```
  {
    "MessageType": "Request",
    "Format": "binary",
    "Subformat": "image/jpeg",
    "Content": b'\xff\xd8\xff\xe0...',  // JPEG binary
    "Submessages": [
      {
        "Label": "description",
        "Format": "text",
        "Subformat": "en",
        "Content": "Process this image for defects"
      }
    ]
  }
```

CBOR encoding allows this entire object to be transmitted compactly.

Example 3: text fallback (JSON over WebSocket text frame)

```
{
  "MessageType": "Request",
  "Format": "binary",
  "Subformat": "audio/wav;base64",
  "Content": "UklGRngAAABXQVZFZm10IBAAAAABAAEAESsAACJWAAACABAAZGF0YYAA...",
  "Submessages": [
    {
```

```
      "Label": "transcription",
      "Format": "text",
      "Subformat": "en-US",
      "Content": "What's the current stock price of Tesla?"
    }
  ]
}
```

8 Message handling and framing

Feature	CBOR Frame	Text Fallback
Frame Type	Binary	Text
Encoding	CBOR	UTF-8 JSON
Binary Data Support	Native (byte string)	Base64 in JSON
Compression Support	Optional (via permessage-deflate)	Optional
Streaming Support	Chunking via multiple frames	Limited

9 Session management (optional)

- Each message MAY include a session ID in "MessageType" or custom field.
- Use WebSocket heartbeat for liveliness checks.
- Session states can be managed using a separate "Control" message.

10 Error handling

- If CBOR decoding fails, server SHOULD:
  - Send back a NLIP error message using the fallback endpoint.
  - Example:

```
{
 "MessageType":"Error",
  "Format": "text",
  "Subformat": "English",
  "Content": "CBOR decoding failed. Fallback to text recommended."
}
```