Ecma/TCXY/<year>/NN

1st Draft **Standard** ECMA-XXX

Edition / Date

Natural Languge Interaction Protocol

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

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 in-telligent 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, we need a standard common protocol that is used widely among interacting agents. This specification proposes such a protocol which would ensure interoperability among various services that use AI based technology.

This document describes the specification of the protocol. The binding of the protocol to underlying base transfer protocols are described in companion documents.

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

Specification of Natural Language Interaction Protocol

1 Scope

This Standard defines the specifications of Natural Language Interaction Protocol (NLIP), which is an application level communication protocol defined between generative AI Agents.

2 Conformance

A conforming implementation of NLIP must provide and support all types of message and submessages and their semantics as described in this specification.

A conforming implementation of NLIP must select a base transfer protocol and conform to the specifications in the binding document of that specification.

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.

ECMA-404, *The JSON Data Interchange Syntax*, 2nd edition (December 2017)

4 Terms and definitions

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

4.1
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. REST over HTTPS is an example of a base transfer protocol.

4.2
JSON message
JSON formatted message is any data that is written to be conformant to ECMA-404 specifcations.

4.3
Data modality
The modality of data refers to the type of information the data is representing. Common examples of data modality include natural language (expressed in human languages), structured text (e.g. JSON, XML, or a computer include language like ECMAScript), video, audio, binary data, sensor data and GPS data representing location coordinates.

4.4
Multi-modal data
A multi-modal Data is data that consists of one or more modalities of data. A multimodal data may have messages that carry only one modality of data, with different messages carrying different modalities. A multimodal data may also have a message which contains of more than one modality.

4.4
End-point
An end-points is a computer system or computer software that communicates using NLIP with other computer systems or computer software, on the same or other machine.

4.5
Agent
A agent is a software which is capable of processing multimodal information, take actions and generate responses based on such processing using an AI model such as a large neural network. The action and response generation may or may not involve a human in the loop.

4.6
Client agent
A client agent is a agent which initiates communication with another agent by sending data to the the latter.

4.7
Server agent
A server agent is a agent which waits for other agents to initiate communication with it.

4.8
Middle agent
A middle agent is an agent which acts like a server agent to an end-point and as a client agent to another end-point.

4.8
Proxy agent
A proxy agent is an agent which acts as a client-agent on behalf of another agent

4.8
Reverse proxy agent
A reverse proxy agent is an agent which acts like a server agent on behalf of another server agent.

5 NLIP Message Formats

In this section, we describe the format of NLIP messages and the required action of the end-point on the receipt of each type of message of sub-message.

Each message exchange via NLIP is formatted as a JSON message. Each message consists of a first submessage and an optional list of other submessages.

5.1 First Submessage

The first submessage in a NLIP is a JSON object consisting of the following fields:

5.1.1 MessageType field

The MessageType field is an optional JSON String value. The value of 'control' is reserved and indicates a message that is used to negotiate control parameters. Control messages are intended to perform operations related to the non-functional aspect of an end-point, e.g. to query about its applicable policies, or to negotiate any parameters associated with the end-point configuration or protocol configuration. Other messages are intended to perform operations related to the function of the end-point. Capitalization is not relevant to the value of the field.

When this field is missing, the server may assume that the message is a data message. Any label other than Control is considered a data message, and its interpretation is left to the end-point receiving the message.

5.1.2 Format field

The format field is required. It is used to specify the format in which the contents field is represented. It is described in Section 5.3.

5.1.3 Subformat field

The subformat field is required. It is used to specify the subcategory of representation within the format. It is described for various formats in Section 5.3.

5.1.4 Content Field

This content field is required. It includes the actual content that is being sent between the client and the server. It must be represented as specified by the format and subformat field.

5.1.5 Submessages Field

This is an optional field. When present, it must be represented as a JSON array of one or more submessages. Each submessage is described in Section 5.2. Following ECMA-404 specifcations, a JSON array is an ordered list of values: the order of the submessages in the Submessages field is significant.

5.2 Submessage

The submessage is a JSON structure consisting of the following fields.

5.2.1 Label Field

The Label field is optional. When present, it must have the key of “Label” without regard of the capitalization of the key. Its value field must be a JSON string. Its example uses could be to carry its submessage’s sequence number or the “role” information in a LLM chat.

5.2.2 Format field

The format field is required. It is used to specify the format in which the contents field is represented. It is described in Section 5.3.

5.2.3 Subformat field

The subformat field is required. It is used to specify the subcategory of representation within the format. It is described for various formats in Section 5.3.

5.2.4 Content Field

This content field is required. It includes the actual content that is being sent between the client and the server. It must be represented as specified by the format and subformat field.

5.3 Format and Subformat field values

The format field must have one of the values described in Table 1. The format key is “format” regardless of case. When subformat is shown within angular brackets <…>, the content inside angular is a placeholder for generic value as described in Notes.

Table 1 — List of Allowed format fields

JSON Value	Subformat Values	Notes
text	<lang>	The content contains natural language text. The subformat identifies the language that is being used for content.
token	<prefix>_<suffix>	The content is an opaque token interpreted by end-points. The subformat begins with a prefix and may contain _ after the prefix and a suffix. Two prefixes are reserved namely ‘authentication’ and ‘conversation. See Section XXX for details on these.
structured	json, uri, xml, html, <prog>	The content is structured content. It can be json, uri, xml or html. <prog> can be name of any programming language. If the name of a programming language is specified, content field contains code in that programming language. If the programming language is not understood/supported by the end-point receiving it, it should send back a response containing an error field.
binary	<content>/<encoding>	The subformat field must have content as one of the values of audio, image, sensor, or generic. The encoding must specific content encoding, e.g. bmp or jpeg for audio, mp3/mov for video etc. A sender may include a . as prefix for encoding. Valid values for subformat include are audio/bmp, audio/.bmp, video/.mp3, generic/.zip etc.
location	text, GPS	The content specifies location. The location may be specified as a text description, or a GPS location.
error	code, text	When the subformat is code, the content includes an error code, which can be a JSON number or a JSON string. When it is text, it is the generic description of an error.
generic	<ext>	The subformat should include the name of an extension that both client and server support. This allows private clients and servers to define their own private format fields.

All conforming end-points must support the processing of all fields, and support at least the text format field with the subformat of ‘English’.

6 Mandatory Exchanges

The following behaviors are required to be supported by an end-point in response to the receipt of various fields.

6.1 Initial Request

The initial request must be made by a NLIP client agent to an NLIP server agent. The NLIP client agent must send an NLIP message including the first submessage. The NLIP server agent must send a response to the NLIP client agent in response to the request.

6.2 Requests containing Token field

If any NLIP message containing a submessage that includes the token format with the subformat beginning with ‘conversation’ is received, the end-point must include the same submessage with the exact same content and subfield in its response message to the peer end-point, except when the conversation token was created by the end-point.

Either the client agent or the server agent may be the first one to create a token field with the subformat field starting with ‘conversation. The other agent can add another submessage with the format of ‘token’ and the subformat of ‘conversation.

An end-point including the token format may choose to end its identity afther the ‘_’ character to identify who originated the conversation submessage. The contents of the conversation submessage are opaque to the protocol and can be interpreted only by the end-point creating it.

6.3 Requests containing Control field

If a NLIP End-Point receives a message which has the control field set to true, it must respond with a control field set to true.

6.4 Requests for redirection

Base64 encoding may not always be desirable. If a client has a large binary content that it wants to transmit using a mechanism that is more efficient than JSON Base64 encoding, it must use a base transfer protocol that supports such a mechanism. The redirection request is done by sending two NLIP messages.

The first message is a request from a client agent to the server agent asking for a redirection. This must be a message with the control field set to True, and the first message should include a text format request for the location to send large uploads.

The response to this request message from the server agent should be a control message containing a submessage with the structured format and subformat of a URI. The URI should be to an end-point where the base transfer protocol can be used to transmit the large binary content.

6.5 Requests for Authentication

A client agent may sent a message to the server agent without authentication. A client agent or server agent may ask the other agent for authentication information. Authentication requests must be sent using a NLIP message with the control field set to ‘true’. The other agent must respond to this request and all subsequent requests with a submessage that contains a field of ‘token’ with the subformat of ‘authentication’. The content of the authentication submessage may be obtained using the mechanisms of the base transfer protocol.

7 Base Transfer Protocol

A base transfer protocol must satisfy the following properties:

7.1 Encrypted Communication

The base transfer protocol must support communication using an encrypted transport. During development of protoypes, the base transfer protocol may be configured to skip encryption. However, any deployed implementation of the NLIP protocol must happen over a communication channel that uses encryption.

7.2 Authentication

The base transfer protocol must implement or enable authentication among the client and server. While the exchange of NLIP messages can happen without authentication between the client and server, the server or the client can ask for authentication. The mechanism for authentication may happen using any existing authentication scheme supported by the base transfer protocol.

7.3 Communcation End Point

The base transfer protocol must support at least one communication end point for the NLIP server agent to use to receive NLIP messages.

A base transfer protocol may support additional end-points to provide for more efficient communication such as upload of large binary content.

Annex A

JSON Specification

The following is the language independendent definition of the NLIP message:

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "definitions": {
    "AllowedFormats": {
      "type": "string",
      "enum": ["text", "token", "structured", "binary", "location", “error”, "generic"],
      "description": "The values of the format field that are defined by NLIP Specification"
    },
    "ReservedTokens": {
      "type": "object",
      "properties": {
        "auth": {
          "type": "string",
          "enum": ["authorization"],
          "description": "Authentication token"
        },
        "conv": {
          "type": "string",
          "enum": ["conversation"],
          "description": "Conversation id token"
        }
      },
      "description": "Reserved values of subformat when token is format, as defined by NLIP Specification"
    },
    "NLIP_SubMessage": {
```

```
    "type": "object",
    "properties": {
      "format": {
        "$ref": "#/definitions/AllowedFormats"
      },
      "subformat": {
        "type": "string",
        "description": "The subformat of the sub-message"
      },
      "content": {
        "oneOf": [
          { "type": "string" },
          { "type": "object" }
        ],
        "description": "The content of the message. Can be a string or a dictionary."
      }
    },
    "required": ["format", "subformat", "content"],
    "description": "Represents a sub-message in the context of the NLIP protocol."
  },
  "NLIP_BasicMessage": {
    "type": "object",
    "properties": {
      "control": {
        "type": "boolean",
        "description": "Whether the message is a control message or not"
      },
      "format": {
        "$ref": "#/definitions/AllowedFormats"
      },
      "subformat": {
        "type": "string",
        "description": "The subformat of the sub-message"
      },
      "content": {
        "oneOf": [
          { "type": "string" },
          { "type": "object" }
        ],
        "description": "The content of the message. Can be a string or a dictionary."
      }
    },
    "required": ["control", "format", "subformat", "content"],
    "description": "The Basic Message with no sub-messages in the context of the NLIP protocol."
  },
  "NLIP_Message": {
    "type": "object",
    "properties": {
      "control": {
        "type": "boolean",
        "description": "Whether the message is a control message or not"
      },
      "format": {
        "$ref": "#/definitions/AllowedFormats"
      },
      "subformat": {
        "type": "string",
        "description": "The subformat of the sub-message"
      },
      "content": {
          "oneOf": [
            { "type": "string" },
            { "type": "object" }
          ],
          "description": "The content of the message. Can be a string or a dictionary."
        },
        "submessages": {
          "type": "array",
          "items": { "$ref": "#/definitions/NLIP_SubMessage" },
          "description": "List of sub-messages"
        }
      },
      "required": ["control", "format", "subformat", "content"],
      "description": "The Message that can include sub-messages in the context of the NLIP protocol."
    }
  },
  "type": "object",
  "properties": {
    "AllowedFormats": { "$ref": "#/definitions/AllowedFormats" },
    "ReservedTokens": { "$ref": "#/definitions/ReservedTokens" },
    "NLIP_SubMessage": { "$ref": "#/definitions/NLIP_SubMessage" },
    "NLIP_BasicMessage": { "$ref": "#/definitions/NLIP_BasicMessage" },
    "NLIP_Message": { "$ref": "#/definitions/NLIP_Message" }
  }
}
```

Annex B
(informative)

Example Interactions

The following section shows examples of NLIP exchanges for an application use-case of <TBD>

Bibliography (if any)

[1] Experimental statistics, US National Bureau of Standards Handbook 91, 1963

[2] Applied Regression Analysis, Draper and Smith, Wiley Edition 2

[3] Statistical Methods for Reliability Data, Meeker, Escobar, 1998, John Wiley & Sons Inc.