MDDP - Metadata Data Decoupling Protocol

Paolo Ippolito - 2023 - version 1.0

I. Introduction

The architectures of web applications since the 90s have evolved becoming more and more complex: from the simple client-server paradigm passing to 3-tier applications up to SOA and microservices architectures.

During the evolution of architectures, web protocols have also evolved, but always using the HTTP protocol as a basis, a protocol created for communications between client and server and adapted in the following years for server-2-server communications such as e.g. SOAP messages of SOA architectures or JSON messages of REST architectures.

However, the use of the HTTP protocol brings with it a functional overhead, such as e.g. redirect mechanism, or restrictions, such as e.g. the management of parallel calls and the non-asynchronicity of the protocol, which do not make the communication between backend services present in the same data center, or often even in the same network, optimal.

Furthermore, the data sent each time has as informative content both the information and what describes its meaning, both the data and the metadata; despite the sending of metadata in the messages exchanged, over the years there has always been the need to create some document that would define the interfaces, such as e.g. the WSDL or OpenAPI YAML interfaces.

II. MDDP

A. *Definition*

MDDP (Metadata Data Decoupling Protocol) was born with the aim of creating a protocol suitable for server-2-server communications, reducing the payload of the data sent, making it possible to send and receive data both in synchronous mode and in asynchronous, and completely separating data from metadata.

The MDDP application protocol uses TCP as the transport layer protocol in its insecure form; a study is underway for the use of MDDP in secure mode (MDDPS) which should both be able to use TLS and support some other form of encryption with offline key exchange.

The protocol consists mainly of two parts:

- The contract, which represents the data exchange interface between client and server and contains all the metadata.
- The message, which represents the information content exchanged between client and server.

III. Contract

A. *Definition*

The contract represents the interface exposed by the server.
The contract consists of:

- Contract Version – unsigned numeric
- Charset used for string representation
- Function List

B. *Function List*

The function represents the single service exposed by the server.
The function consists of:

- Function name
- Function description
- Function Versions

C. *Function Versions List*

The function version represents the single version of the single service exposed by the server.
The function version consists of:

- List of request message fields
- list of response message fields

D. *Field*

The field represents the data that is sent within the message.
It is made up of:

- Field Name
- Field Description
- Field Length, more than 0 if it have a fixed length, -1 for variable length
- Field Data Type, it could be:
  Binary
  String
  Integer
  Float
  Long
  Double
  Boolean
  Complex
  List
  Dictionary
- Subfields: present only if the Data Type is of the Complex, List or Dictionary type, it contains the subfields that represent the data type

E. *Complex Data Type*

The data type Complex represents a complex object and is used in the MDDP protocol as the primary data type of the

content of the body or in fields where the use of a complex object is expected.
It is made up of:

- Bitmap Length (short unsigned int) represented in bytes length
- Bitmap – byte 0…N: bitmap of the fields present, for each bit it represents the presence (value equal to '1' or less (value equal to '0') of the sub-field
- Subfields list

F. *List Data Type*

The Data Type List is an object that contains a list of objects of a single data type that is represented in the only subfield present in the contract.

G. *Dictionary Data Type*

The Data Type List is an object that contains a list of key objects - value of two data types that are represented in the two subfields present in the contract.

IV. Message

The Message represent the data exchanged by client and server and consists of a 16 byte long header and a body containing the information content of the message.

A. *Header*

The MDDP message header is a 16 byte long binary buffer, which contains the following data:

- Message Length – 4 byte unsigned integer: represents the length of the entire message
- Protocol Version – 1 byte unsigned integer: Represents the protocol version, currently the default value of '1'
- Status Flag - 1 byte binary: Contains a bitmap that represents various information about the status of the message; only bit 0 is valued in the request, the rest is valued in response by the server
  - 0° bit - Message type: '1' if it is a response, '0' if it is a request
  - 1° bit - warning obsolete Contract version: it is set to '1' by the server if a contract version less recent than the one managed by the server is sent in the request
  - 2° bit - warning obsolete Function version: it is set to '1' by the server if a function version less recent than the one managed by the server is sent in the request
  - 3° bit - unknown Contract version: it is set to '1' by the server if a contract version not known by the server is sent in the request
  - 4° bit - unknown Function version: it is set to '1' by the server if a function version not known by the server is sent in the request
  - 5° bit: unknown Function Number: it is valued at '1' by the server if a function number not known by the server is sent in the request
  - 6° bit: message format error: it is set to '1' by the server if the request message is not formatted correctly
  - 7° bit: unknown Protocol version: it is set to '1' by the server if a protocol version not known by the server is sent in the request
- Contract Version - 2 byte numeric: version of the contract used to generate the message
- Function Number - 2 byte numeric: number of the function called to process the message
- Function Version - 2 byte numeric: version number of the function called to process the message
- Correlation Key - 4 byte numeric: correlation key used to match the request and response

B. *Body*

The body is composed of an object of type Complex which represents the message and contains the data which is interpreted through the contract with the selection of the version of the function.

V. Communication workflow

The message is sent from client c to server s.

The Server s reads the first 4 bytes of the header to be able to recover the length n of the message and then the next n-4 bytes to be able to recover the whole message. Subsequently it processes the next 12 bytes to verify that the header is formally correct and that it contains correct references to the contract, function and version of the function; in the event that more updated versions are present, it sets the warning flags of the obsolete versions in the response header, and in the event that the versions are not compatible with what is reported in the contract, it replies configure the error status flags and echoes the body received.
Subsequently it begins to process the message and to extract the request/response fields based on what is reported in the contract; in the event of a non-formally correct message, the status flag bit of the format error is raised and the received body is echoed.

VI. Conclusions

The version of this protocol is born to optimize the communication between server processes, but it could be more optimizable in the future with the contribution of the Open Source community.

Appendix

A. *Fixed field length*

There are represented the field length for all the types that are not ignoring length in contract:

Integer	4 bytes
Float	4 bytes
Long	8 bytes
Double	8 bytes
Boolean	1 byte

B. Variable field length

The variable field length is an unsigned int, represented by 2 byte before the value of the field.
It could be used by this data types:

- Binary
- String
- Complex
- List
- Dictionary

License

References

[1] Java implementation of MDDP: https://gitlab.com/ippolito/mddp
[2] RFC 2616 - Hypertext Transfer Protocol -- HTTP/1.1: https://www.rfc-editor.org/rfc/rfc2616