Monday, 22 Juni 2020

ISO8583 DAN JSON

by :
Ferisa Tri Putri Prestasi

About writer

- *Mathematics College Student'16/8th semester ITS Surabaya*
- *Lab Assistant Computation Mathematics ITS Surabaya*
- *Head of Research Development and Knowledge Management Data Science Indonesia Region Jawa Timur*

linkedin.com/in/ferisaprestasi

github.com/prestasicode

ISO8583

(International Organization for Standardization)

Glimpse of ISO first,

The **International Organization for Standardization** (ISO) is an international-standard setting body composed of representatives from various national standard Organization. In contrast to many international organizations, which utilize the British English form of spelling, the ISO uses English with Oxford Spelling as one of its official languages along with French and Russian.

Source : https://en.wikipedia.org/wiki/International_Organization_for_Standardization

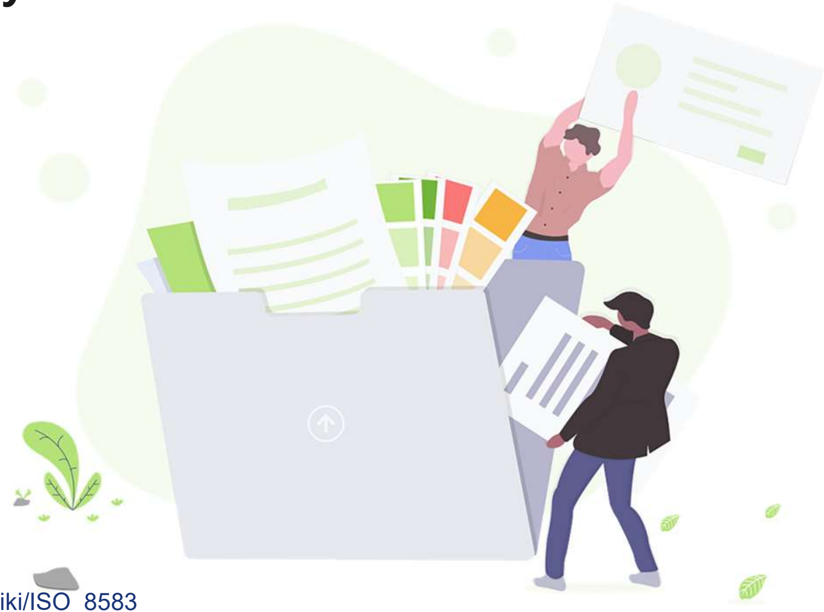
Why???

Because : financial transaction

ISO 8583

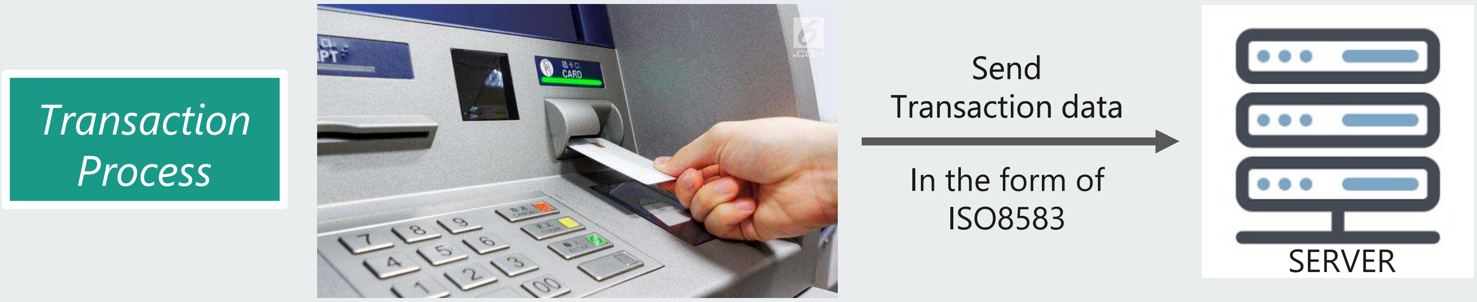
ISO 8583 is an international standard for **financial transaction** card originated interchange messaging. It is the International Organization for Standardization standard for systems that exchange electronic transactions initiated by cardholders using **payment cards**.

ISO 8583
ISO 8583 adalah sebuah format data yang diakui
dan digunakan secara internasional guna transaksi keuangan



Source : https://en.wikipedia.org/wiki/ISO_8583

ISO8583 as one of the standard (protocols) various kind of data interchange



Transaction Process

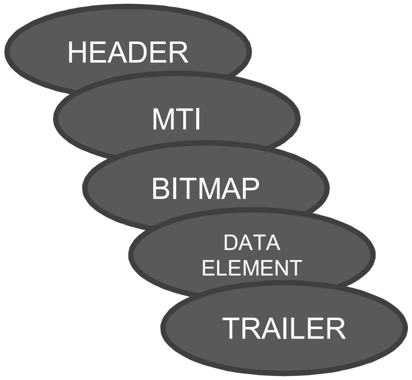
Send Transaction data

In the form of ISO8583

SERVER

Source : https://www.codeproject.com/Articles/100084/Introduction-to-ISO

MESSAGE STRUCTURE OF ISO8583



MESSAGE HEADER

- Used as an initial marker of message.
- Envelopes the application data and are used for routing and message integrity.

APPLICATION DATA

consist of ISO message including **Message Type Indicator** (MTI), **BIT MAP** (indicating which data elements are present) and **ISO Data Element** (the fields of the message).

MESSAGE TRAILER

- Used as a marker for end messages or differentiators between messages.
- Envelopes the application data and are used for routing and message integrity.

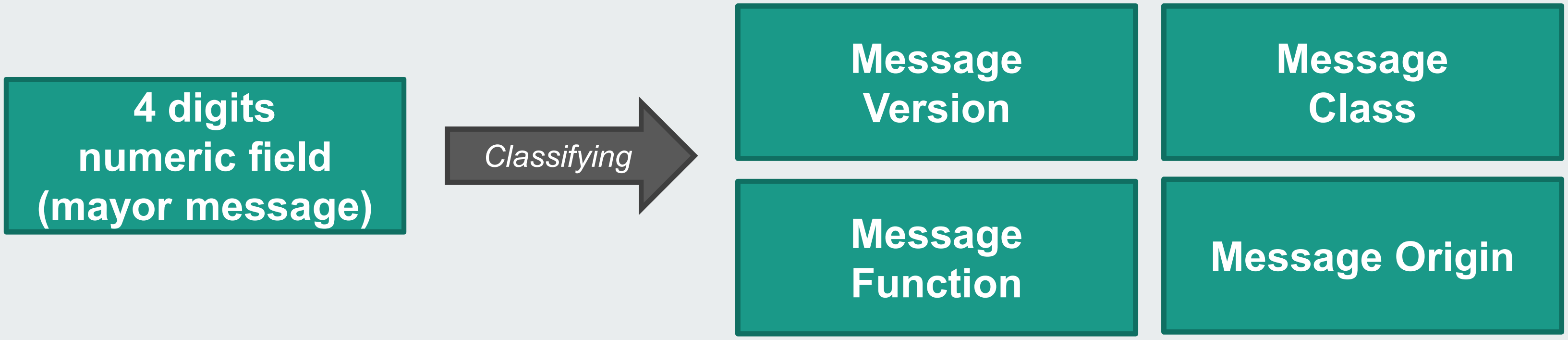
Source : https://www.codeproject.com/Articles/100084/Introduction-to-ISO

Details Component ISO8583

Message Type Indicator

BITMAP

1 MTI (*Message Type Indicator*)



4 digits numeric field (mayor message)

Classifying

Message Version

Message Class

Message Function

Message Origin

Let's discuss *MIT* one by one >>>>

Message Version

“Different versions of the ISO 8583 standard are indicated in the first position of the Message Type Indicator.“

Code	Meaning
0xxx	*ISO 8583:1987 version*
1xxx	*ISO 8583:1993 version*
2xxx	*Reserved for ISO use*
3xxx	*Reserved for ISO use*
4xxx	*Reserved for ISO use*
5xxx	*Reserved for ISO use*
6xxx	*Reserved for ISO use*
7xxx	*Reserved for ISO use*
8xxx	*Reserved for national use*
9xxx	*Reserved for private use*

Source : https://www.codeproject.com/Articles/100084/Introduction-to-ISO

Message Class

function as a type of message, for example whether it is in the form of requests, responses, advice and others - others. The following is a table for the message function.

Code	Meaning	Usage
x0xx	*Reserved by ISO*	
x1xx	*Authorization Message*	*Determine if funds are available, get approval but don't post to account for reconciliation*
x2xx	*Financial Message*	*Determine if funds are available, get approval and post to account for reconciliation*
x3xx	*File Actions Message*	*Used for hot-card, TMS and other exchanges*
x4xx	*Reversal & Chargeback Message*	*Reversal (x4x0 or x4x1): Reverses the action of a previous authorization* *Chargeback (x4x2 or x4x3): Charge back a previously cleared financial message*
x5xx	*Reconciliation Message*	*Transmits settlement information message*
x6xx	*Administrative Message*	*Transmits administrative advice. Often used for failure message (e.g message reject or failure to apply)*
x7xx	*Fee Collection Message*	
x8xx	*Network Management Message*	*Used for secure key exchange, logon, echo test, and other network functions*
x9xx	*Reserved for ISO use*	

Source : https://www.codeproject.com/Articles/100084/Introduction-to-ISO
https://rizkimufrizal.github.io/belajar-iso-8583/

Message Function

has function to defines the purpose of the message to be sent, here is a table for the message class.

Code	Meaning	Notes
xx0x	For a request, requiring approval	
xx1x	For a response to a request	
xx2x	For an advice of an action that has already been taken, not requiring approval but still requiring a response	
xx3x	For a response to an advice	
xx4x	For notification	
xx5x	Reserved for ISO use	
xx6x	Instruction	*ISO8583:2003 only*
xx7x	Instruction acknowledgement	
xx8x	Reserved for ISO use	*Some implementation use + acknowledgement*
xx9x	Reserved for ISO use	*Some implementation use - acknowledgement*

Source : https://www.codeproject.com/Articles/100084/Introduction-to-ISO
https://rizkimufrizal.github.io/belajar-iso-8583/

Message Origin

Has a function to define the source of sending data, for example such as the acquirer (the financial institution that issues the buyer's card), the issuer (the financial institution that deals directly with the EDC seller's machine) and others.

Code	Meaning
xxx0	Acquirer
xxx1	Acquirer repeat
xxx2	Card issuer
xxx3	Card issuer repeat
xxx4	Other
xxx5	Other repeat
xxx6	Reserved for ISO use
xxx7	
xxx8	
xxx9	

Source : https://www.codeproject.com/Articles/100084/Introduction-to-ISO
https://rizkimufrizal.github.io/belajar-iso-8583/

GLIMPSE EXAMPLE OF MTI

1101

have a meaning ***Authorization Request*** where :

- 1 -> ISO 8583 version 1993
- 1 -> Authorization message
- 0 -> Request
- 1 -> Acquirer repeat

0110

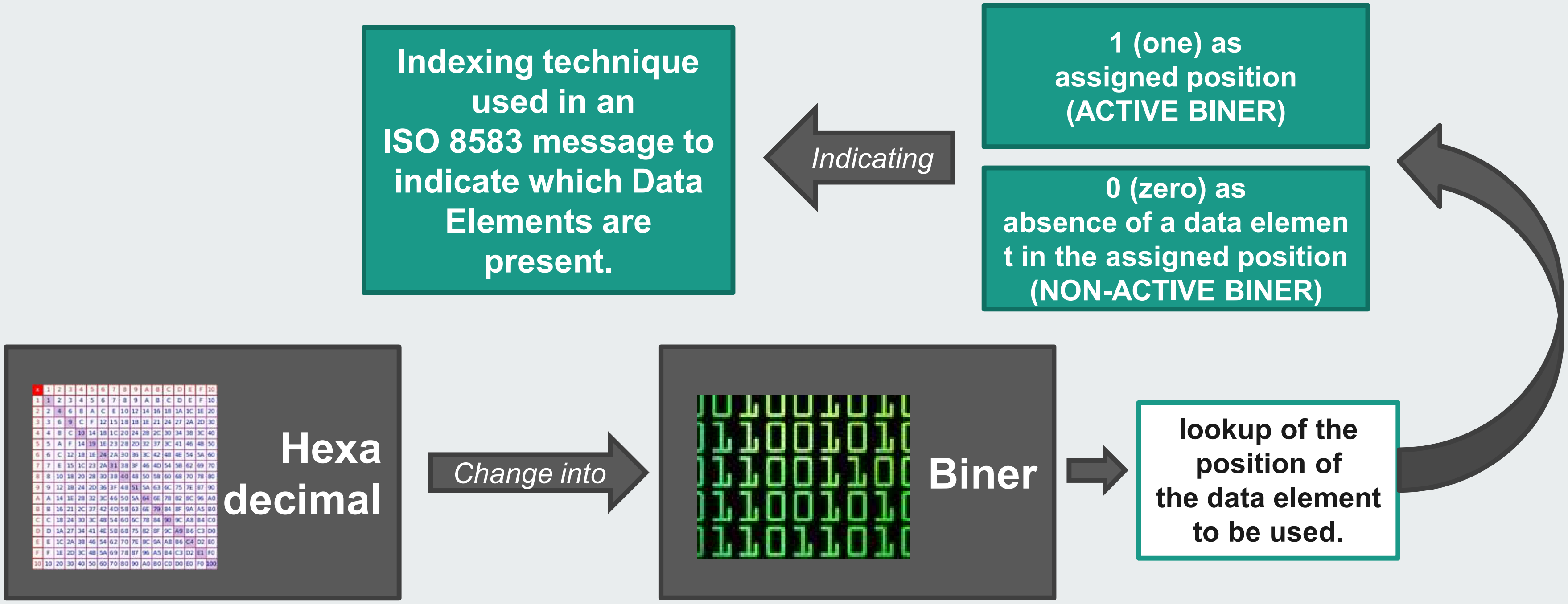
example that Responses **0110** artinya ***Authorization Response*** where :

- 0 -> ISO 8583 version 1987
- 1 -> Authorization message
- 1 -> Request response
- 0 -> Acquirer

2 BITMAP

Each application transaction includes one (1) bit map.

- consists of 64 bits numbered from the left starting with bit 1 (one)
- If any ISO message does not support secondary bit map processing, then the first bit of the bit map is '0'.



Indexing technique used in an ISO 8583 message to indicate which Data Elements are present.

Indicating

1 (one) as assigned position (ACTIVE BINER)

0 (zero) as absence of a data elemen t in the assigned position (NON-ACTIVE BINER)

Hexa decimal

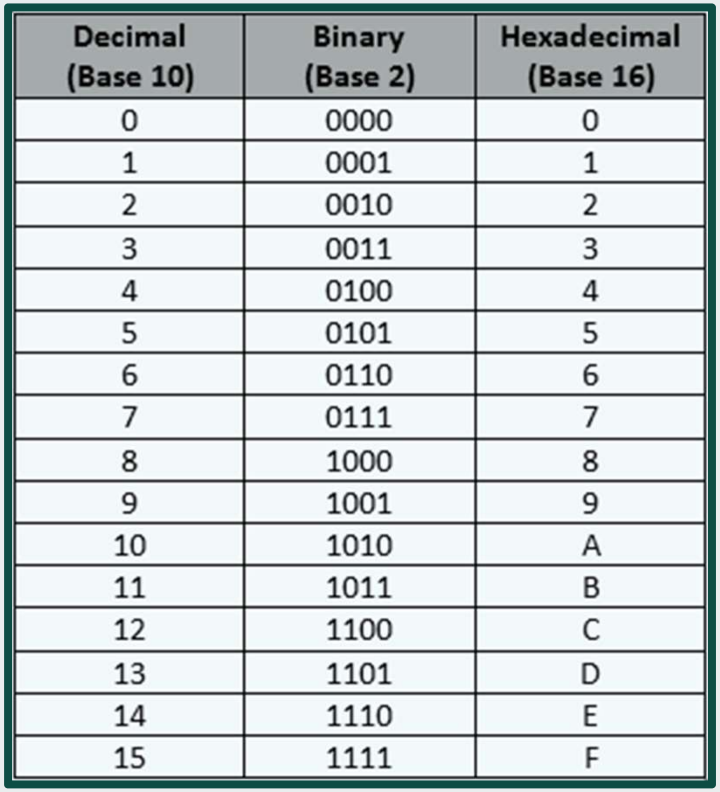
Change into

Biner

lookup of the position of the data element to be used.

CONVERSION TABLE

from Hexadecimal to Biner



Decimal (Base 10)	Binary (Base 2)	Hexadecimal (Base 16)
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

A message can contain up to three bitmaps in the latest version of the ISO 8583 standard. The bitmap may be transmitted as 8 bytes (binary), or sometimes with the 8 bytes unpacked into 16 hexadecimal characters 1-9, A-F (ASCII).

EXAMPLE :

F23C449108E080000000000000000000021

where the above bitmap consists of **32 characters** it can be ascertained that the above bitmap is a **secondary bitmap.**
How to change the above hexadecimal to binary is first break the hexadecimal to 2 digits then convert 1 digit hexadecimal to binary as follows.

Next slide for detail example >>>

3 TYPES OF BITMAP

1. PRIMARY BITMAP

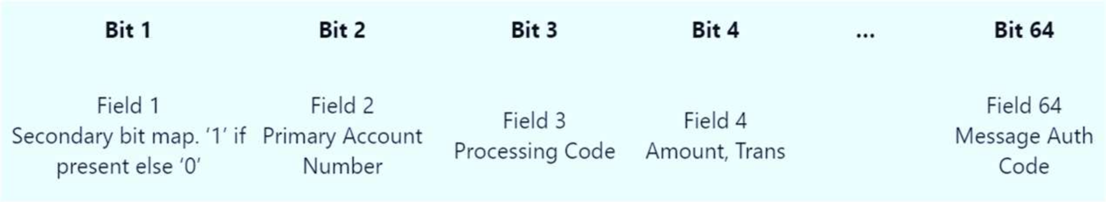
A message contains at least one bitmap called the Primary Bitmap, indicating the presence of **Data Elements 1 up to 64.**

2. SECONDARY BITMAP

A Secondary Bitmap may be located at Data Element one, and hence the first bit in Primary Bitmap tells us whether there is a secondary bitmap or not. The secondary bitmap indicates the presence of **Data Elements 65 up to 128.**

3. TERTIARY BITMAP

The secondary bitmap indicates the presence of **Data Elements 129 up to 192.**



Bit 1	Bit 2	Bit 3	Bit 4	...	Bit 64
Field 1 Secondary bit map. '1' if present else '0'	Field 2 Primary Account Number	Field 3 Processing Code	Field 4 Amount, Trans		Field 64 Message Auth Code

Source : https://www.codeproject.com/Articles/100084/Introduction-to-ISO

Hexa	Hexa	Biner	Biner
F	2	1111	0010
3	C	0011	1100
4	4	0100	0100
9	1	1001	0001
0	8	0000	1000
E	0	1110	0000
8	0	1000	0000
0	0	0000	0000
0	0	0000	0000
0	0	0000	0000
0	0	0000	0000
0	0	0000	0000
0	0	0000	0000
0	0	0000	0000
0	0	0000	0000
2	1	0010	0001

Combine all biner number like this

111100100011110001000100100100010
0001000111000000100000000000000000
0000000000000000000000000000000000
00000000000000000000000000100001

From the binaries above it can be seen that the all **number 1 shows that the active binaries,**
the following are the active binaries in position: 1,2,3,4,7,11,12,13,14,18 ,22,25,28,32,37 , 41.42,43,49,123 and 128.

Data Element

- Data Elements are fields carrying the information of the transaction itself.
- There are up to 128 Data Elements in the original ISO 8583 (1987) standard, and up to 192 Data Elements in later releases.
- Each Data Element has a specified meaning and format.

Data Element Format

ISO-8583 specifies different kind of fields that basically fall in following categories:

- ❑ Fixed length
  - Numeric
  - Alphanumeric
  - Binary
- ❑ Variable length
  - Max-length 99
    - ➢ Numeric
    - ➢ Alphanumeric
    - ➢ Binary
- ❑ Max-length 999
  - Numeric
  - Alphanumeric
  - Binary
- ❑Nested Message

FIELD TYPE ATTRIBUTES

Abbreviation	Meaning
a	Alphabetic, including Blanks
n	Numeric Values only
s	Special Characters only
an	Alphanumeric
as	Alphabetic & Special Characters only
ns	Numeric and Special Characters only
ans	Alphabetic, numeric & Special Characters
b	Binary Data
z	Tracks 2 & 3 code set as defined in ISO 7811 & ISO 7813
h	Hex Data
LL, LLL	Length of variable field that follows. ‘LL’ - Two-digit length indicator (1 byte BCD) ‘LLL - 3-digit length indicator (2 bytes BCD)
..17 ..125	Variable field of up to 17. The ‘..’ is a two-digit length indicator (1 byte BCD) specifying the number of digits defining the length of the variable data to follow. Variable field of up to 125 characters. The ‘…’ is a three-digit length indicator (2 bytes BCD), specifying the number of digits defining the length of the variable data to follow.

#	Name	Value	Hex Value
0	MTI	0800	08 00
1.a	PRIMARY BITMAP	Indicates presence of secondary bitmap plus fields 3, 11 and 41	A0 20 00 00 00 80 00 10
1.b	SECONDARY BITMAP	Indicates presence of field 70	04 00 00 00 00 00 00 00
3	PROCESSING CODE	000000	00 00 00
11	SYSTEM TRACE AUDIT NUMBER	000001	00 00 01
41	CARD ACCEPTOR TERMINAL IDENTIFICATION	29110001	32 39 31 31 30 30 30 31
60	RESERVED FOR PRIVATE USE	TEST MESSG	00 10 54 45 53 54 20 4D 45 53 53 47
70	NETWORK MANAGEMENT INFORMATION CODE	301	03 01

In above sample, two new fields #60 and #70 are present.

Here is our message representation:

Message:

```
0800A020 00000080 00100400 00000000
 00000000 00000001 32393131 30303031
 00105445 5354204D 45535347 0301
```

MTI: 0800
Primary bitmap: A0200000 00800010
Secondary bitmap: 04000000 00000000
Field 03: 000000
Field 11: 000001
Field 41: 3239313130303031
(ASCII for "29110001")
Field 60: 0010 54455354204D45535347
(length=10, value="TEST MESSG")
Field 70: 0301

Let's break down this bitmap >>>

RESULT BITMAP EXPLANATION

PRIMARY BITMAP

Byte	Hex Value	Bit Value	Field #
0	A0	1010 0000	Secondary bitmap present plus #3
1	20	0010 0000	11
2	00	0000 0000	
3	00	0000 0000	
4	00	0000 0000	
5	80	1000 0000	41
6	00	0000 0000	
7	10	0001 0000	60

SECONDARY BITMAP

Byte	Hex Value	Bit Value	Field #
0	04	0000 0100	70
1	00	0000 0000	
2	00	0000 0000	
3	00	0000 0000	
4	00	0000 0000	
5	80	0000 0000	
6	00	0000 0000	
7	00	0000 0000	

JSON

(JavaScript Object Notation)

Glimpse of JSON

But, wait.. What is metadata? Next slide >>>

- Accessing Metadata is often possible through services offered by the provider and can be retrieved in a structured format that could include raw text, XML (eXtensible Markup Language, or in this example **JSON.**
- **JavaScript Object Notation** is an open standard file format, and **data saving and interchange format**, that uses human-readable text to store and transmit data objects consisting of attribute-value pairs and array data types (or any other serializable value).
- **JSON** is programming language independent (JavaScript is not required to use it).
- **JSON** is based on the object literal notation of JavaScript (emphasis on the word "notation").
- **JSON** represents data in a way that is friendly to universal programming concepts.

Source : https://en.wikipedia.org/wiki/JSON

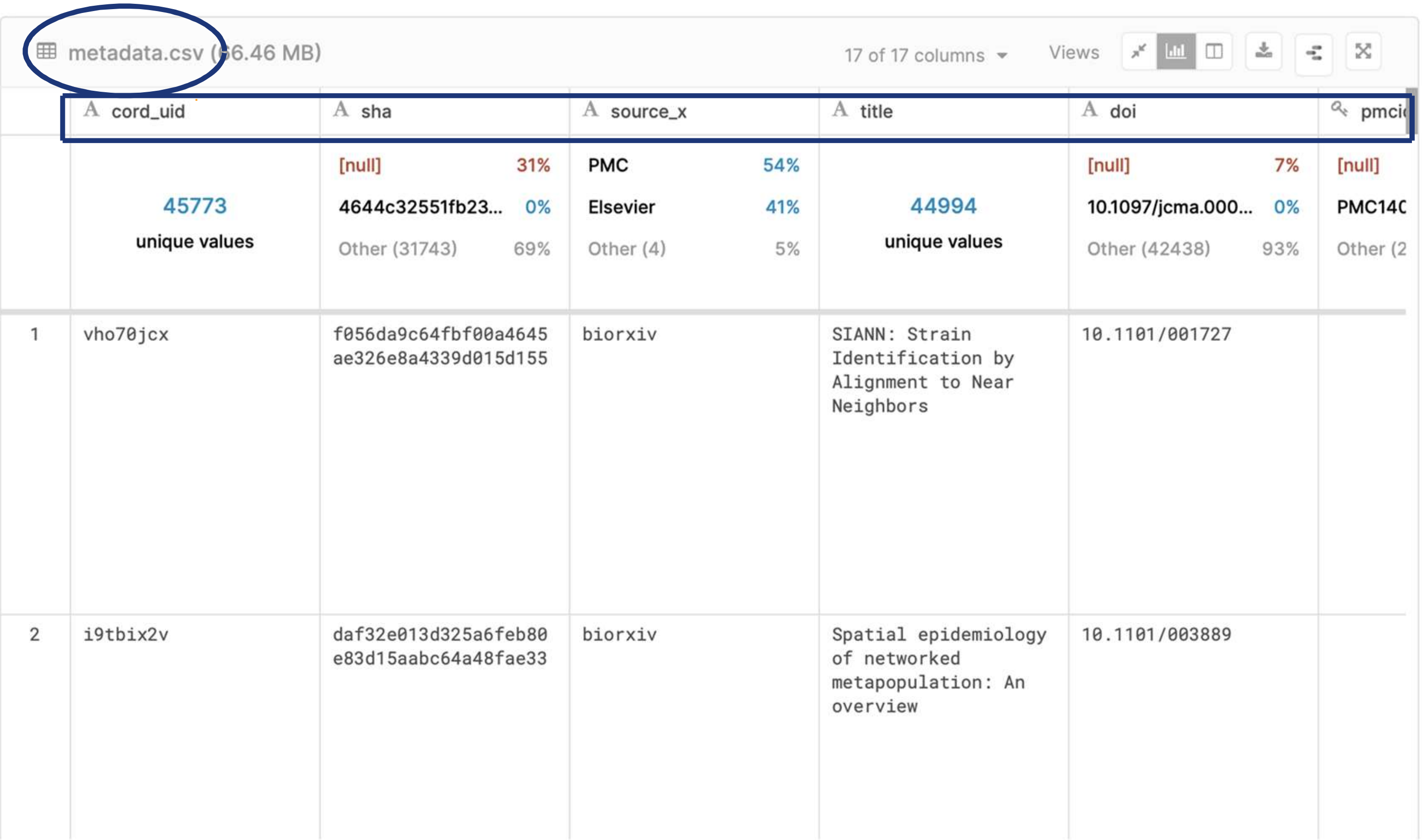
Keyword (Additional Slide)

Glimpse of Meta-data

- **Meta-data** is information about the physical data, technical and business processes, data rules and constraints, and logical and physical structures of the data, as used by an organization.
- **Meta-data** is a set of data that describes and gives information about other data.

Source : Mosley, Brackett, Early, & Henderson, 2010; Oxford Dictionaries

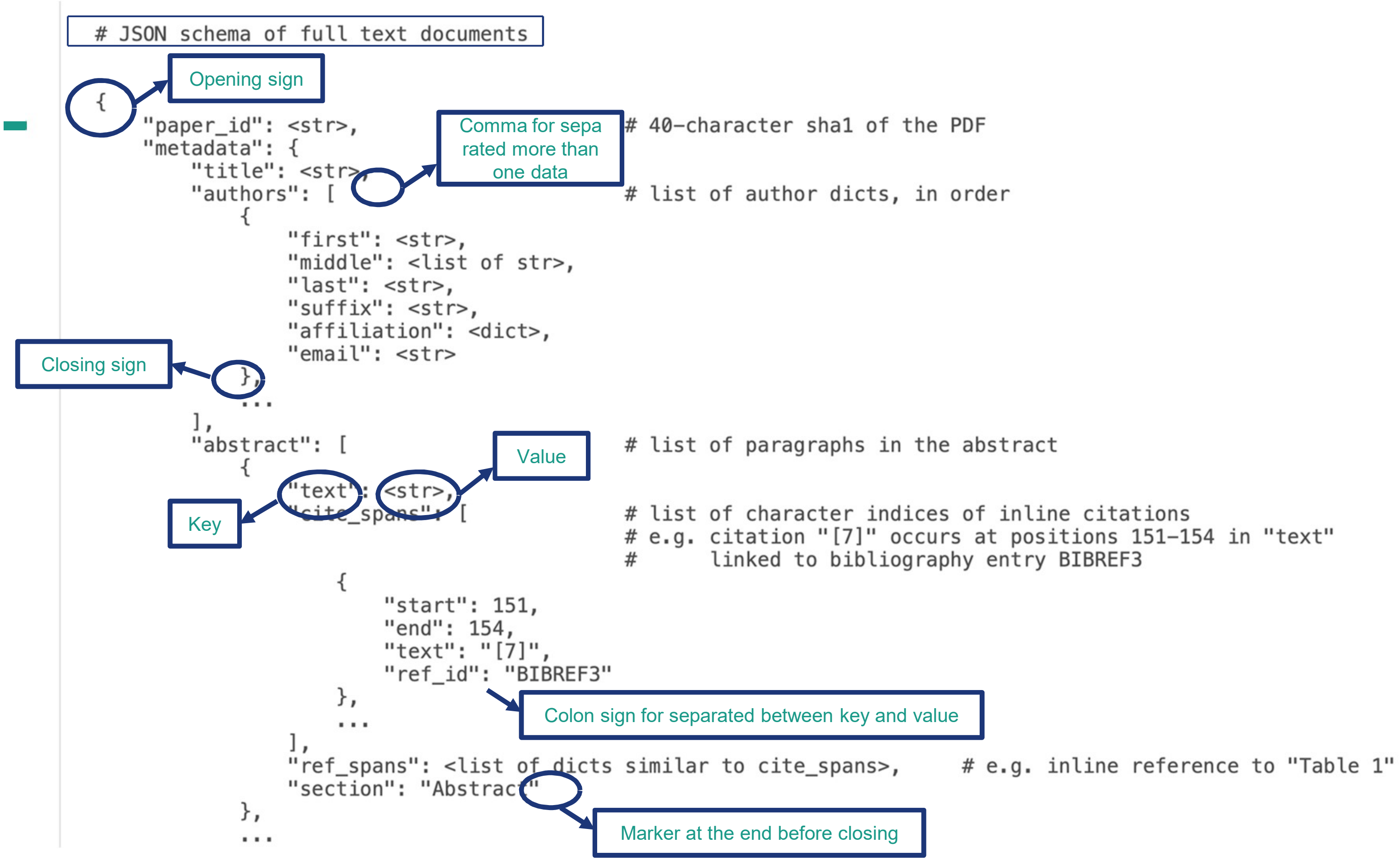
When developing a database structure for an information system, there needs to be a metadata schema.



metadata.csv (56.46 MB)

	cord_uid	sha	source_x	title	doi	pmci
	45773 unique values	[null] 31% 4644c32551fb23... 0% Other (31743) 69%	PMC 54% Elsevier 41% Other (4) 5%	44994 unique values	[null] 7% 10.1097/jcma.000... 0% Other (42438) 93%	[null] PMC14C Other (2
1	vho70jcx	f056da9c64fbf00a4645ae326e8a4339d015d155	biorxiv	SIANN: Strain Identification by Alignment to Near Neighbors	10.1101/001727	
2	i9tbix2v	daf32e013d325a6feb80e83d15aabc64a48fae33	biorxiv	Spatial epidemiology of networked metapopulation: An overview	10.1101/003889	

JSON Structure



```
# JSON schema of full text documents
{
    "paper_id": <str>,                          # 40-character sha1 of the PDF
    "metadata": {
        "title": <str>,
        "authors": [                            # list of author dicts, in order
            {
                "first": <str>,
                "middle": <list of str>,
                "last": <str>,
                "suffix": <str>,
                "affiliation": <dict>,
                "email": <str>
            },
            ...
        ],
        "abstract": [                           # list of paragraphs in the abstract
            {
                "text": <str>,
                "cite_spans": [                 # list of character indices of inline citations
                                                # e.g. citation "[7]" occurs at positions 151-154 in "text"
                                                #      linked to bibliography entry BIBREF3
                    {
                        "start": 151,
                        "end": 154,
                        "text": "[7]",
                        "ref_id": "BIBREF3"
                    },
                    ...
                ],
                "ref_spans": <list of dicts similar to cite_spans>,     # e.g. inline reference to "Table 1"
                "section": "Abstract"
            },
            ...
```

Opening sign

Comma for sepa rated more than one data

Closing sign

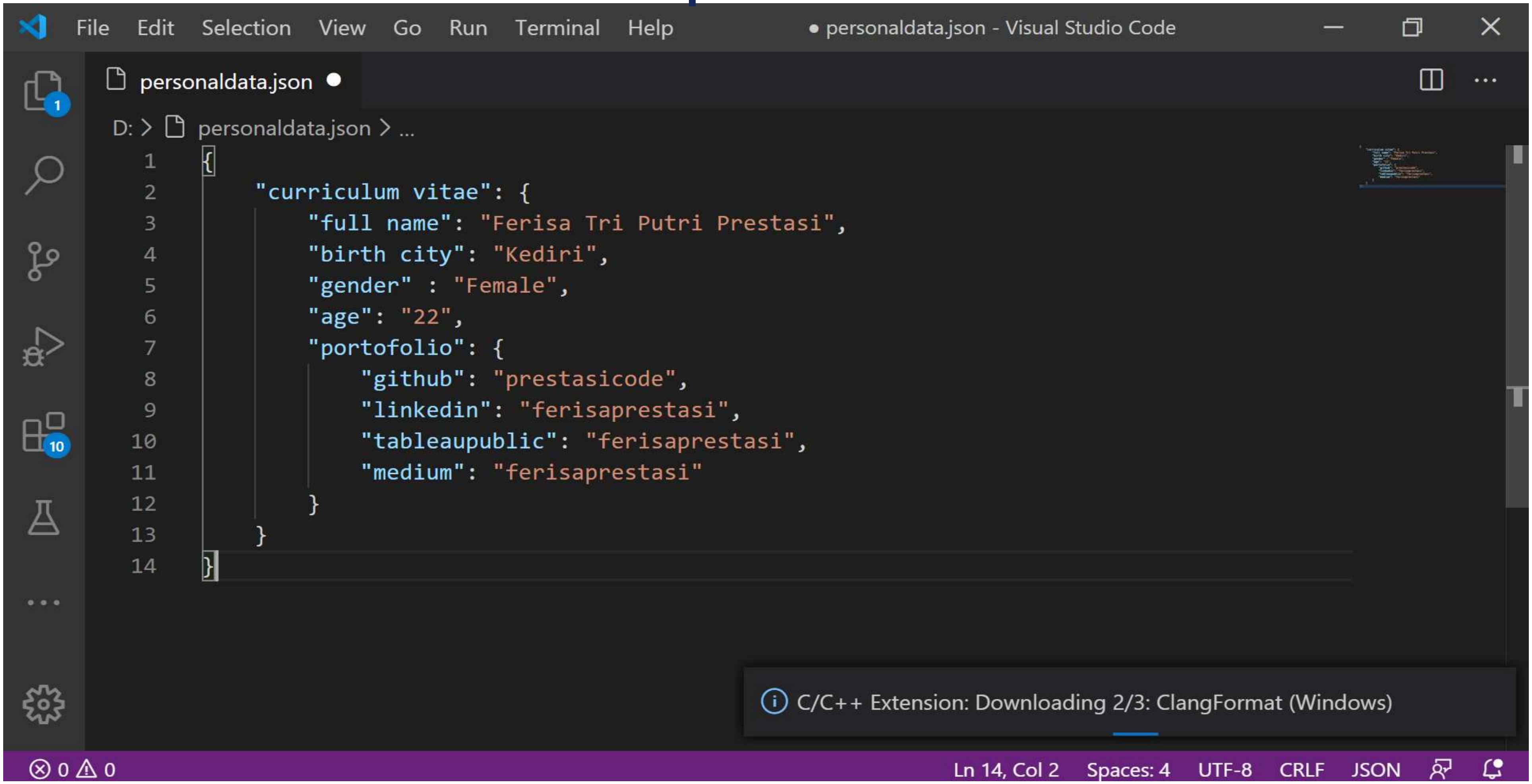
Value

Key

Colon sign for separated between key and value

Marker at the end before closing

Example of JSON

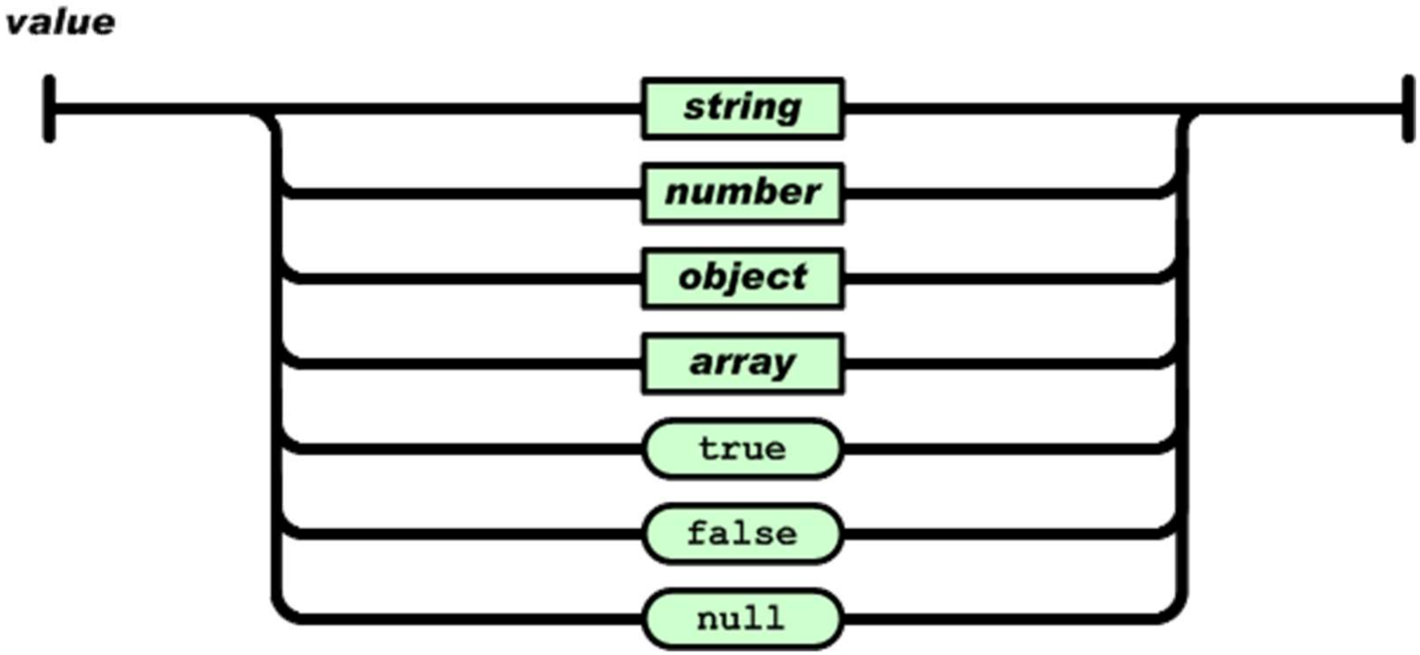


File
Edit
Selection
View
Go
Run
Terminal
Help
● personaldata.json - Visual Studio Code
personaldata.json ●
D: > personaldata.json > ...
1
10

```
{
    "curriculum vitae": {
        "full name": "Ferisa Tri Putri Prestasi",
        "birth city": "Kediri",
        "gender" : "Female",
        "age": "22",
        "portofolio": {
            "github": "prestasicode",
            "linkedin": "ferisaprestasi",
            "tableaupublic": "ferisaprestasi",
            "medium": "ferisaprestasi"
        }
    }
}
```

C/C++ Extension: Downloading 2/3: ClangFormat (Windows)

JSON Compatible Value Format



Source : https://www.petanikode.com/json-pemula/

Thank you

Ferisa Tri Putri Prestasi

06111640000031

ferisatri04@gmail.com