Published in The plain and simple series

anupamvarghese Follow
Jan 14, 2016 · 6 min read · Listen

ISO 8583. An introduction. Plain and Simple

This post is dedicated to all who have just stepped into the financial transaction processing technology world as we know it and want a primer on one of the most prolific protocols powering this world- the ISO 8583.

Introduction

Almost all of us would have swiped a card or two at an ATM or a PoS terminal (or a Square dongle ;)). At the very least, the card serves as an identity factor. Among other things , the magnetic stripe on each card stores something called the PAN (Primary Account Number). For most credit cards, it is the same as the credit card number printed on the front surface/ plastic.

The act of swiping a card on a card reader essentially involves passing this 'identity' of the card to the electronic sub-system and represents a **Card Present** type of transaction. Alternatively, one could have typed in this information (card number) on a screen of an online shopping interface but then this would effectively become a **Card Not**

you get confused by 'debit' and 'credit': Debit = Deduct from your card/ account. Credit = Create money in your card/ account. In case you wish to find out more about the Latin origins of the words, start here:Why do accountants use debits and credits instead of simple pluses and minuses?)

Origin

Way back in 1987 (I think), The International Organization for Standardization (ISO) declared a standard called the 8583 to facilitate the flow of transaction information interoperably. I believe Visa and Mastercard had come into existence much before this and some form of interoperability existed even before this standard was declared. The important point is that both Visa and Mastercard had adopted this standard at some point in time. Also, the standard has gone through numerous iterations and various financial institutions have tweaked it to create many flavors/ variants.

What and Why?

So what is ISO 8583? It is one of the many standards describing how to pack certain data fields such that it could reliably be unpacked as well and is mostly relevant for the financial transaction processing world.

So this standard helps the electronic system which reads the card number, the transaction amount and other relevant data fields to pack it all up so that it could be transmitted electronically to a transaction processing system where it could then be unpacked back into individual data components and then processed. It also helps the transaction processing system pack and send the response back to the initiating device where it could again be unpacked and the customer be intimated of the transaction response.

There exist numerous methods for packing and unpacking data. It could be as simple as comma separated fields. Eg: I could choose to send the transaction information as simple comma separated values as:
"1234123412341234,1000,INR,987" (Card Number, Amount, Currency, Merchant ID).

The issue with such a simplistic model of data packing is that it lacks meta information.

important to consider that the packing and unpacking could be coded easily into mainframes, not sure about this one.

Many folks have already begun writing obituaries to the ISO 8583 protocol thanks to the advent of the younger and dynamic (but not leaner) ISO 20022. However, thanks to its proliferation, ISO 8583 will be a difficult one to get rid of soon and hence one way or the other, in this industry you will need to know this veteran.

Principles

The ISO 8583 message is based on the principles that:
a. In a transaction message, you only get to pick any number of fields from a predefined set of fields. So, if you need a field called 'My girlfriend's phone number', sorry, ain't possible.
b. The meta information of which fields are present in the message are also a part of the message payload in a data structure called the 'bitmap'.

Structure

Most implementations contain a few bytes dedicated to a fixed header (eg: ^A^TISO016000010) after which the actual ISO 8583 message starts.

MTI

The Message Type Indicator.
The first 4 bytes describe the message type. Eg:

02 00
which tells that the message is actually a financial transaction request. (The response to this request would also be in ISO 8583 and would carry an MTI: 02 10). Various MTIs exist and can be found on the web.

Bitmap

Now that we know that this is a financial transaction, we would naturally expect a few

Imagine a switchboard with 64 ON/OFF switches arranged one after the other from left to right. Lets assume that each switch represents each of the 64 main pre-defined fields. (The 1st field is interesting, we will come to that later). For every field that is present in the message, assume that we turn that particular switch ON and for every field that is absent, we ensure that the switch at that position is turned OFF.

For example, assume we had only one field present and if that field was field no. 3, all other switches except the third one from the left would be in OFF position.

If we write 1 for every switch that is ON/ field that is present and 0 for every switch that is OFF/ field that is not present, we get a series of 1s and 0s. This series of 1s and 0s is called the binary bitmap. It is, as I'd mentioned, a linear visual map of which all fields are present in the message payload.

Data Element Map, B64. Binary. 64 bits

Eg:
F2 38 80 01 08 E0 80 0F

11110010 00111000 10000000 00000001 00001000 11100000 10000000 00001111

(all the bit positions that are 1 implies the corresponding fields are present)
Hex Binary (Positions that have 1)
F2= 11110010 -> (1,2,3,4,7)
38= 00111000 -> (11,12,13)
80= 10000000 -> (17)
01= 00000001 -> (32)
08= 00001000 -> (37)
E0= 11100000 -> (41,42,43)
80= 10000000 -> (49)
0F= 00001111 -> (61,62,63,64)

Bingo, we've just read the map! Therefore the fields that will be present in this message

 

message contains another bitmap with another 64 bits.

Extended bitmap, b64. Binary 64 bits

80 00 00 00 00 00 00 00
(=hex .extended bitmap field)
(80)10000000 -> (position 64+1=65)

This extended bitmap shows that field number 65 is also present in this message.

Data elements

Immediately after the bitmap, the data elements start serially. From the bitmap we know that fields 2,3,4,7 are present one after the other. All that we need to do is to read them one by one. Each field number has a predefined type in the ISO 8583 definition and has a predefined length. Some fields have variable length in which case the first N bytes provide the length of the field.

Example:

Data Element 2. Length 16. Value : 0000011319353459 = Primary account number
Data Element 3. Length 6. Value : 011000 =Processing code. 011000 = cash withdrawal
Data Element 4. Length 12. Value : 000000020000 =Amount 200.00
Data Element 7. Length 10. Value : 0804030013 =DateTime DDMMhhmmss
Data Element 11. Length 6. Value : 051028 =Systems Trace number
Data Element 12. Length 6. Value : 083013 =Time, hhmmss
Data Element 13. Length 4. Value : 0804 =Date, MMDD
Data Element 17. Length 4. Value : 0804 =CaptureDate, MMDD
Data Element 32. Length 6. Value : 123456 =Acquiring institution ID code 123456
Data Element 37. Length 12. Value : 192165102801 =Retrievel Ref. No.
…
Data Element 65. Length 50. Value : Customer Withdrawal =Statement narrative, right pad spaces.

many more layers involved in actual implementations of the protocol. The following lines could help you learn further.

Additional resources

1.The Wikipedia on ISO8583

2. jPOS — an open source implementation started by Alejandro Revilla and a default choice for many developers

3. (paid) ISO specs: 2003

Originally published at www.anupamvarghese.com.