Transaction Specs – Custom Pay API - EPI EPX Payment Interface

Public - Security Level 0

November 2022

Date	Version	Technical Writer	SME
05/15/19	1.0	J. Kelley	Kent Glenn; Kevin Carpenter
7/2/21	1.1	M. Billips	Bringing up to date
4/12/22	1.2	M. Billips	Add BRIC storage endpoints
7/6/2022	1.3	M. Billips	Update JSON to XML Mapping table
9/29/2022	1.4	M. Billips	Update taxExempt tag to a string with possible values of N or Y

Contents

Table of Figures

EPI Overview

This guide contains information on mandatory and optional fields for the following types of transactions commonly used within the EPX Payment Interface (EPI):

- ACH
- Authorization
- Batch lookup
- BRIC Storage
- Refund
- Reverse
- Sale
- Void
- PIN Debit

Field Types

The following sections provide examples for each transaction type. The mandatory fields will be listed in **bold**.

- To ensure a successful response, mandatory fields need to be submitted with the transaction request.
- Optional fields may be omitted; however, optional fields might still be used during the transaction. For example, if optional AVS fields are provided, they are validated and an AVS response is sent back. Also, some optional fields can be made mandatory in a merchant's profile to comply with Risk and Underwriting requirements.

Transaction Types

Each transaction type below contains an example of the endpoint and the method (GET, PUSH, POST, etc.). If required, it will also contain an example of any JSON payload for the call. For additional information on industry specific transaction types and tags, reference the respective EPX transaction specifications along with the XML mapping tag names in the EPX Data Dictionary.

ACH

The ACH sale or Account Debit transaction is used to debit funds from a bank account.

ACH Sale / Debit Example – Checking Account

```
POST /ach
```

The /ach endpoint requires **Account, Routing Number, Amount, Transaction,** and **Address** with **First and Last Name** shown below in the ACH transaction payload.

```
{
 "account": "9876543210",
 "routingNumber": 012345678,
 "amount": 127.99,
 "transaction": 123,
 "batchID": 20201103,
 "address": {
   "firstName": "Galileo",
   "lastName": "Galilei"
 },
 "standardEntryClass": "PPD"
}
```

AVS

The account verification transaction is used to validate a customer's account information before running a financial transaction. This is done by sending a $0.00 amount during the transaction.

Account Verification Example

```
POST /avs
```

The /avs endpoint requires **Account, Expiration Date, Transaction ID, Card Entry Method,** and **Industry Type** in the payload. Optional tags that will help with AVS and validation are Address, Zip Code, and CVV2 as shown below in the Ecommerce transaction payload.

```
{
  "account": "4111111111111111",
  "expirationDate": "2512",
  "transaction": 123,
  "industryType": "E",
  "cardEntryMethod": "E",
  "batchID": 201217,
  "address": {
     "firstName": "Galileo",
     "lastName": "Galilei" ,
     "address": "123 Main Street",
     "state": "DE",
     "zipCode": "12345"
  },
  "cvv2": "123"
}
```

Batch

- Lookup (totals, etc.)

```
GET /batch/{id}
```

- Close all batches

```
PUT /batch
```

- Close specific batchID

```
PUT /batch/{id}
```

Batch Close Example

```
http://xxx.xxxxxx.com/batch/123

Method: PUT
```

The request requires a **Transaction ID.**

```
{
  "transaction": 123
}
```

Sale

The sale transaction is an authorization and capture within the same transaction. Because of this, the authorization is immediately captured by the EPX platform so no additional transaction is required to capture the authorization. If the sale is approved, the transaction will close and settle during the next batch close time for funding to take place.

The sale transaction in JSON appears as one of the two examples below:

```
POST /sale
POST /sale/{bric}
```

Sale Examples

Sale Example – Sale Authorization (Auth) and Capture

```
http://xxx.xxxxxx.com/sale/

Method: POST
```

The /sale endpoint requires **Amount, Account, Expiration Date, Batch ID, Transaction ID, Industry Type,** and **Card Entry Method**. Optional tags that will help with AVS and validation are Address, Zip Code, and CVV2 as shown below in the Ecommerce transaction payload.

```
{
  "amount": 179.00,
  "account": "4111111111111111",
  "expirationDate": "2512",
  "batchID": 201217,
  "transaction": 123,
  "industryType": "E",
  "cardEntryMethod": "E",
  "address": {
     "firstName": "Galileo",
     "lastName": "Galilei" ,
     "address": "123 Main Street",
     "state": "DE",
     "zipCode": "12345"
  },
  "cvv2": "123"
  }
```

Sale Example – Sale with BRIC

The BRIC is being used to reference a previous credit card transactions GUID / Token in the EPX system. Since the EPX BRIC is a unique reference value, there is no need to include the account number and expiration date. If the address and zip code was supplied with the original transaction for the BRIC in use it will be included with the transaction request.

```
http://xxx.xxxxxx.com/sale/{BRIC}

Method: POST
```

The /sale/{bric} endpoint requires **Amount, Batch ID, Transaction ID, Industry Type,** and **Card Entry Method** as shown below in the Ecommerce transaction payload.

```
{
  "amount": "127.99",
  "batchID": 201217,
  "transaction": 123,
  "industryType": "E",
  "cardEntryMethod": "Z"
}
```

Sale Example – Sale Authorization Only (no capture)

The authorization only transaction is an authorization that will hold funds equal to the AMOUNT of the transaction on a cardholders' account. A subsequent capture transaction is required in order for settlement and funding to take place.

```
http://xxx.xxxxxx.com/sale

Method: POST
```

The /sale authorization only (no capture) requires the **capture** tag as a Boolean set to false along with **Amount, Account, Expiration Date, Batch ID, Transaction ID, Industry Type,** and **Card Entry Method**. Optional tags that will help with AVS and validation are Address, Zip Code, and CVV2 as shown below in the Ecommerce transaction payload.

Note: Capture flag by default is set to true if not supplied in the payload.

```
{
  "amount": 179.00,
  "account": "4111111111111111",
  "expirationDate": "2512",
  "batchID": 201217,
  "transaction": 123,
  "capture": false,
  "industryType": "E",
  "cardEntryMethod": "E",
  "address": {
     "firstName": "Galileo",
     "lastName": "Galilei" ,
     "address": "123 Main Street",
     "state": "DE",
     "zipCode": "12345"
  },
  "cvv2": "123"
}
```

Sale Example – Sale Edit (Auth and Capture)

The Edit Sale (Authorization and Capture) transaction is used to edit an open Sale (Authorization and Capture) transaction. The transaction is a BRIC-based request and can only be performed on an approved Sale transaction that has not been voided, closed, or settled.

```
http://xxx.xxxxxx.com/sale/{BRIC}

Method: PUT
```

```
{
 "amount": 131.99,
 "tipAmount": 5.55
}
```

Sale Example – Incremental Authorization

The incremental authorization transaction is used to authorize an additional amount of funds to add to the original authorization only transaction amounts. The transaction is a BRIC-based request and can only be performed on an approved authorization only transaction that has not been reversed, voided, or captured. To add an additional amount of money to an authorization only transaction, or incrementally authorize a new amount, the original authorization only BRIC is required.

Note: It is strongly recommended to reference the EPX Incremental Authorization Transaction Specs to ensure accuracy of implementation.

```
http://xxx.xxxxxx.com/sale/inc/{BRIC}

Method: PUT
```

The /sale/inc/{bric} endpoint requires **Amount, Batch ID, Transaction ID, Industry Type,** and **Card Entry Method** as shown below in the Ecommerce transaction payload.

The amount is only the additional amount needed, not the new total amount. For example, if the initial authorization was for 20.00 and the new total authorization amount needs to be 30.00, the transaction payload would contain an **Amount** of 10.00.

```
{
  "amount": "10.00",
  "batchID": 201217,
  "transaction": 123,
  "industryType": "E",
  "cardEntryMethod": "Z"
}
```

Response

The response from a /sale transaction will be the same regardless of additional fields included in the request.

Note: For additional response tags to be included in the response, supply "responseLevel": 1 in the payload of the transaction request.


```
{
 "data": {
   "authorization": "009835",
   "response": "00",
   "text": "APPROVAL 009835"
 },
 "errors": null,
 "reference": {
   "bric": "00DGZ9BTJB05MDGU002",
   "timestamp": "2019-05-21T19:38:02Z"
 }
```


Capture

The capture transaction is used to capture a previous authorization only to allow settlement to occur. A capture can be run for an AMOUNT equal or less than the amount of the referenced authorization. If the capture is approved, the transaction will close and settle during the next batch close time in order for funding to take place.

Capture Example – Capture with BRIC

```
PUT /sale/{BRIC}/capture
```

The /sale/{bric}/capture endpoint requires **Amount, Batch ID, Transaction ID, Industry Type,** and **Card Entry Method** as shown below in the Ecommerce transaction payload.


```
{
  "amount": "127.99",
  "batchID": 201217,
  "transaction": 123,
  "industryType": "E",
  "cardEntryMethod": "Z"
}
```

Lookup

```
GET /sale/{bric}
```

Sale Lookup Example

```
http://xxx.xxxxxx.com/sale/0123456789012345678
```

Sale Lookup Example Response


```
{
  "data": {
    "authamount": "50.00",
    "authorization": "008251",
    "authtrandategmt": "12-Apr-2022",
    "batchid": "220409",
    "closeddate": "",
    "epxid/customer": "1234",
    "epxid/dba": "1",
    "epxid/merchant": "1234567",
    "epxid/terminal": "1",
    "folionumber": "",
    "invoicenumber": "",
    "markedtosettle": "N",
    "ordernumber": "",
    "referencenumber": "",
    "rentalnumber": "",
    "response": "00",
    "system/trantype": "CCE1",
    "tipamount": "0.00",
    "transaction": "5",
    "userdata/1": "",
    "userdata/10": "",
    "userdata/2": "",
    "userdata/3": "",
    "userdata/4": "",
    "userdata/5": "",
    "userdata/6": "",
    "userdata/7": "",
    "userdata/8": "",
    "userdata/9": "",
    "void": "N"
  },
  "errors": null,
  "reference": {
    "bric": "00DJHBDLULHUFULK56M",
    "timestamp": "2022-04-12T16:01:29Z"
  }
}
```

Transaction Listing Example

The /sale endpoint provides the ability to get a list of transactions based on the four part key. When called directly, it will list all transactions specific to that four part key at the terminal level. If the detail property is set to 'dba,' it will tell the endpoint to return all transactions at the DBA level, essentially using the first three parts of the four part key.

```
http://xxx.xxxxxx.com/sale
http://xxx.xxxxxx.com/sale?detail=dba
```

Transaction Lookup Example Response


```
{
  "data": {
    "rowCount": "3",
    "rows": [
      {
        "authamount": "50.00",
        "authbric": "09LJHBDJRZLBQLT1GFJ",
        "authtrandategmt": "12-Apr-2022",
        "batchid": "220409",
        "epxid/customer": "1234",
        "epxid/dba": "1",
        "epxid/merchant": "1234567",
        "epxid/terminal": "1",
        "system/trantype": "CCE1",
        "transaction": "5",
        "void": "N"
      },
      {
        "authamount": "50.00",
        "authbric": "09LJHBDZFZMKN8MKKDW",
        "authtrandategmt": "12-Apr-2022",
        "batchid": "220409",
        "epxid/customer": "1234",
        "epxid/dba": "1",
        "epxid/merchant": "1234567",
        "epxid/terminal": "1",
        "system/trantype": "CCE1",
        "transaction": "6",
        "void": "N"
      },
      {
        "authamount": "370.00",
        "authbric": "09LJHBDZVZVJ76TRKDL",
        "authtrandategmt": "12-Apr-2022",
        "batchid": "220409",
        "epxid/customer": "1234",
        "epxid/dba": "1",
        "epxid/merchant": "1234567",
        "epxid/terminal": "1",
        "system/trantype": "CCE1",
        "transaction": "7",
        "void": "N"
      }
    ]
```

```
  },
  "errors": null,
  "reference": {
    "bric": "00DJHBDVD5DPYKKWZKU",
    "timestamp": "2022-04-12T16:06:39Z"
  }
}
```

Ping

```
GET /ping
```

Ping Example

```
http://xxx.xxxxxx.com/ping
```

Response

```
{
  "data": null,
  "errors": null,
  "reference": {
     "bric": "0123456789012345678",
     "timestamp": "2019-05-05T20:53:41Z"
  }
}
```

Refund

The refund or return transaction is a transaction used to return funds to an account previously acted upon by a settled sale or capture transaction. A single refund for the full amount of the original transaction or numerous partial refunds can be performed with dollar amounts less than and not to exceed the total amount of the original sale or capture being acted upon.

Note: Reference the appropriate industry specific Transaction Specs for important information surrounding returns on the EPX platform.

```
POST /refund/{bric}
POST /refund
```

Refund Example 1 – Refund with BRIC

The BRIC is being used to reference the previous sale or capture transaction GUID / Token in the EPX system. Since the EPX BRIC is a unique reference value, there is no need to include the account number and expiration date. If the address and zip code was supplied with the original transaction for the BRIC in use it will be included with the transaction request

```
http://xxx.xxxxxx.com/refund/{BRIC}
```

The /refund/{bric} endpoint requires **Amount**, **Batch ID**, **Transaction ID**, **Industry Type**, and **Card Entry Method** as shown below in the Ecommerce transaction payload.

```
{
  "amount": "127.99",
  "batchID": 201217,
  "transaction": 123,
  "industryType": "E",
  "cardEntryMethod": "Z"
}
```

Refund Example 2 – Refund with account info

```
http://xxx.xxxxxx.com/refund
```

The /refund endpoint requires **Amount**, **Account**, **Expiration Date**, **Batch ID**, **Transaction ID**, **Industry Type**, and **Card Entry Method** as shown below in the Ecommerce transaction payload.

```
{
  "amount": 179.00,
  "account": "4111111111111111",
  "expirationDate": "2512",
  "batchID": 201217,
  "transaction": 123,
  "industryType": "E",
  "cardEntryMethod": "E",
  "address": {
     "firstName": "Galileo",
     "lastName": "Galilei" ,
     "address": "123 Main Street",
     "state": "DE",
     "zipCode": "12345"
  },
  "cvv2": "123"
  }
```

Refund Lookup

```
GET /refund/{bric}
```

Reverse

The reverse transaction is used to remove the authorization hold on a credit card and void the transaction within the same request. This will release the funds that are being held at the issuing bank. The BRIC sent during this transaction is one from an open / unsettled sale or open / uncaptured authorization only transaction.

Note: Reference the appropriate industry specific Transaction Specs for additional information surrounding reversals on the EPX platform.

```
POST /reverse/{bric}
```

Reverse Example

```
http://xxx.xxxxxx.com/reverse/{BRIC}
```

The /reverse/{bric} endpoint requires **Batch ID**, **Transaction ID**, **Industry Type**, and **Card Entry Method** as shown below in the Ecommerce transaction payload.

```
{
  "batchID": 201217,
  "transaction": 123,
  "industryType": "E",
  "cardEntryMethod": "Z"
}
```

Void

The Void transaction is used to stop a sale, capture, or refund transaction prior to settlement. If the transaction has already been settled, this function will no longer be available.

```
PUT /void/{bric}
```

Void Example

```
http://xxx.xxxxxx.com/void/{BRIC}
```

The /void/{bric} endpoint requires **Batch ID**, **Transaction ID**, and **Card Entry Method** as shown below in the Ecommerce transaction payload.

```
{
  "batchID": 201217,
  "transaction": 123,
  "cardEntryMethod": "Z"
}
```

PIN Debit Sale

The PIN debit sale transaction moves the amount specified out of the bank account associated with the debit card account number. Movement of the funds completes once the debit clearing cycle has completed for the day.

```
POST /debit
```

PIN Debit Sale Example

```
http://xxx.xxxxxx.com/debit/
```

The /debit endpoint requires **Amount, Track Data, PIN Block, Batch ID, Transaction ID, Card ID,** and **Card Entry Method** as shown below in the transaction payload.

```
{
  "amount": 179.00,
  "trackData": "4000000000000002=25121010000000000000?",
  "pinBlock": "2DD951BAD77CAD53FFFF9876543210E0001A",
  "batchID": 2107070,
  "transaction": 123,
  "cardID": "P",
  "cardEntryMethod": "D",
  "address": {
     "firstName": "Galileo",
     "lastName": "Galilei" ,
     "address": "123 Main Street",
     "state": "DE",
     "zipCode": "12345"
  },
  "cvv2": "123"
  }
```

PIN Debit Refund

The PIN debit refund transaction will return the amount specified to the bank account associated with the debit card account number. This transaction is used to return funds if the debit clearing cycle has completed for the associated batch.

```
POST /debit/refund
```

PIN Debit Refund Example

```
http://xxx.xxxxxx.com/debit/refund/
```

The /debit/refund endpoint requires **Amount**, **Track Data**, **PIN Block**, **Batch ID**, **Transaction ID**, **Card ID**, and **Card Entry Method** as shown below in the transaction payload.

```
{
  "amount": 79.00,
  "trackData": "4000000000000002=25121010000000000000?",
  "pinBlock": "2DD951BAD77CAD53FFFF9876543210E0001A",
  "batchID": 2107070,
  "transaction": 123,
  "cardID": "P",
  "cardEntryMethod": "D",
  "address": {
     "firstName": "Galileo",
     "lastName": "Galilei" ,
     "address": "123 Main Street",
     "state": "DE",
     "zipCode": "12345"
  },
  "cvv2": "123"
  }
```

BRIC Storage

The BRIC Storage transaction presents the capability to create a new BRIC or update an existing one within the EPX system, without interaction with the card networks. This transaction type is available for both credit card (CC) and ACH.

To create a new BRIC for a credit card, the account number and expiration date are required. To create a new BRIC for ACH, both the ACCOUNT_NBR and ROUTING_NBR are required.

When updating an existing BRIC/GUID, the original BRIC / GUID must be specified along with any transaction data you would like to update.

Credit Card BRIC Storage Example

```
POST /storage/bric
```

The /storage/bric endpoint requires **Account**, **Expiration Date**, **Transaction ID**, **Card Entry Method**, and **Industry Type** in the payload. Optional tags that will help with AVS are Address and Zip Code as shown below in the Ecommerce transaction payload.

```
{
  "account": "4111111111111111",
  "expirationDate": "2512",
  "transaction": 123,
  "industryType": "E",
  "cardEntryMethod": "E",
  "batchID": 20220412,
  "address": {
"firstName": "Galileo",
"lastName": "Galilei" ,
```

```
"address": "123 Main Street",
"state": "DE",
      "zipCode": "12345"
  }
}
```

Credit Card Storage BRIC Update Example

```
PUT /storage/bric/{bric}
```

The /storage/bric/{bric} endpoint requires the original storage BRIC in the URL endpoint along with the desired data and API tags that need to be updated such as **Account**, **Expiration Date**, or **ADDRESS** info in the payload.

```
{
  "account": "4111111111111119",
  "expirationDate": "4912",
  "transaction": 123,
  "batchID": 20220412,
  "address": {
"firstName": "Update",
"lastName": "BRIC" ,
"address": "789 Main Street",
"state": "DE",
      "zipCode": "54321"
  }
}
```

Industry Type

Industry Type overview

The EPI API requires the INDUSTRY_TYPE (industryType) tag to be present in the transaction request. This will ensure the transaction is processed with the appropriate industry type. The use of the INDUSTRY_TYPE (industryType) tag allows any Retail, CAT, Banking, ECOM, or MOTO transaction to be sent to the same 4-part key and eliminates the need to use unique TERMINAL_NBR(s) configured specifically for each industry type.

INDUSTRY_TYPE tag

The table below shows the TRAN_TYPE values with the corresponding INDUSTRY_TYPE value mapping and descriptions.

	INDUSTRY_TYPE	
TRAN_TYPE industry code	**INDUSTRY_TYPE (industryType) value mapping**	**INDUSTRY_TYPE (industryType) description**
C, B, R	P	Cardholder Present (Retail, CAT, Banking)
E	E	E-Commerce
M	M	MOTO

- Variable Type: Alpha
- Max Length: 1

Example:

```
"industryType":"P"
```

TLV Data

TLV overview

EPX requires Money Transfer (Visa) and MoneySend (MasterCard) data to be presented in tag-length-value (TLV) format, which is a method of encoding information in a single string, versus multiple EPX tags.

TLV_SETS tag

Figure 1 shows an example TLV_SETS tag that contains TLV sets for "RECEIVER," "SENDER," and "TRAN_DATA."

Figure 1: Example TLV_SETS tag

```
TLV_SETS =
"0036000008RECEIVER001005Steve003005Smith0055000006SENDER001004John003005Jones0110
1654910411722111110023000009TRAN_DATA00100202"
```

The following table describes the data in Figure 1:

TLV tag	Plaintext value	TLV representation
"RECEIVER" TLV set		
000 (Tag 000 is required)	RECEIVER	000008RECEIVER
001	Steve	001005Steve
003	Smith	003005Smith
"SENDER" TLV set		
000 (Tag 000 is required)	SENDER	000006SENDER
001	John	001004John
003	Jones	003005Jones
011	5491041172211111 (This is the Account Number)	01101654910411722111111
"TRAN_DATA" TLV set		
000	TRAN_DATA	000009TRAN_DATA

(Tag 000 is required)		
001	02 (This is the Funding Source; in this case Debit)	00100202

JSON to XML Mapping

The table below contains the JSON to XML conversion. Use this table as reference for transactions and field types.

Table 1. JSON to XML Mapping

GROUP	FIELD	REGEX VERIFICATION	TYPE	XML_TAG
address	address	^([a-zA-Z0-9\/\.\-_,#'&+\s\xc0-\xff]){0,30}$	S	ADDRESS
address	city	^([a-zA-Z0-9\/\.\-_,#'&+\s\xc0-\xff]){0,25}$	S	CITY
address	firstName	^.{0,25}$	S	FIRST_NAME
address	lastName	^.{0,25}$	S	LAST_NAME
address	state	^([a-zA-Z0-9\/\.\-_,#'&+\s]){0,3}$	S	STATE
address	zipCode	^([A-Za-z0-9\-\s]){0,10}$	S	ZIP_CODE
epxId	customer	^([0-9A-Z]){1,16}$	S	CUST_NBR
epxId	dba	^([0-9A-Z]){1,16}$	S	DBA_NBR
epxId	merchant	^([0-9A-Z]){1,16}$	S	MERCH_NBR
epxId	terminal	^([0-9A-Z]){1,16}$	S	TERMINAL_NBR
userData	1	^.{0,80}$	S	USER_DATA_1
userData	2	^.{0,80}$	S	USER_DATA_2
userData	3	^.{0,80}$	S	USER_DATA_3
userData	4	^.{0,80}$	S	USER_DATA_4
userData	5	^.{0,80}$	S	USER_DATA_5
userData	6	^.{0,80}$	S	USER_DATA_6
userData	7	^.{0,80}$	S	USER_DATA_7
userData	8	^.{0,80}$	S	USER_DATA_8
userData	9	^.{0,80}$	S	USER_DATA_9
userData	10	^.{0,80}$	S	USER_DATA_10
	account	^([0-9,*]){4,19}$	S	ACCOUNT_NBR
	aci		S	ACI
	aciExt	^(AE | AF | AO | AS | CA | DS | IA | IP | MI | NS | PS | RA | RB | RS | SA | TR | UP){0,1}$	S	ACI_EXT
	actionCode		S	ACTION_CODE
	addenda1		S	ADDENDA_1
	addenda2		S	ADDENDA_2
	addenda3		S	ADDENDA_3
	addenda4		S	ADDENDA_4
	addenda5		S	ADDENDA_5
	addenda6		S	ADDENDA_6
	addenda7		S	ADDENDA_7
	addenda8		S	ADDENDA_8
	addenda9		S	ADDENDA_9
	addenda10		S	ADDENDA_10
	airlineProcID		S	AIRLINE_PROC_ID

	amount	^(\d{1,12}\.) | (\d{0,4}\.\d{1,8}) | (\d{0,5}\.\d{1,7}) | (\d{0,6}\.\d{1,6}) | (\d{0,7}\.\d{1,5}) | (\d{0,8}\.\d{1,4}) | (\d{0,9}\.\d{1,3}) | (\d{0,10}\.\d{1,2}) | (\d{0,11}\.\d{1,1})$	F	AMOUNT
	amxAdditionalData		S	AMX_ADDITIONAL_DATA
	arrivalDate		D	ARRIVAL_DATE
	authAccountNumber		S	AUTH_ACCOUNT_NBR
	authAmount		F	AUTH_AMT
	authAmount2		F	AUTH_AMOUNT
	authAmountRequested		F	AUTH_AMOUNT_REQUESTED
	authAvailableBalance		F	AUTH_AVAILABLE_BAL
	authAVS		S	AUTH_AVS
	authBric	^([0-9A-Za-z]){19,20}$	S	AUTH_GUID
	authCardA		S	AUTH_CARD_A
	authCardB		S	AUTH_CARD_B
	authCardC		S	AUTH_CARD_C
	authCardD		S	AUTH_CARD_D
	authCardE		S	AUTH_CARD_E
	authCardF		S	AUTH_CARD_F
	authCardG		S	AUTH_CARD_G
	authCardH		S	AUTH_CARD_H
	authCardI		S	AUTH_CARD_I
	authCardJ		S	AUTH_CARD_J
	authCardK		S	AUTH_CARD_K
	authCardL		S	AUTH_CARD_L
	authCardM		S	AUTH_CARD_M
	authCardN		S	AUTH_CARD_N
	authCardO		S	AUTH_CARD_O
	authCardType		S	AUTH_CARD_TYPE
	authCurrencyCode		S	AUTH_CURRENCY_CODE
	authCVV2		S	AUTH_CVV2
	authEBTBalanceCash		F	AUTH_EBT_BAL_CASH
	authCardCountryCode		S	AUTH_CARD_COUNTRY_CODE
	authCardCountryName		S	AUTH_CARD_COUNTRY_NAME
	authCardCurrencyCode		S	AUTH_CARD_CURRENCY_CODE
	authCardCurrencyName		S	AUTH_CARD_CURRENCY_NAME
	authEBTBalanceFood		F	AUTH_EBT_BAL_FOOD
	authEMVData		S	AUTH_EMV_DATA
	authFee		F	AUTH_FEE
	authFeeText		S	AUTH_FEE_TEXT
	authLedgerBalance		F	AUTH_LEDGER_BAL
	authMaskedAccountNumber		S	AUTH_MASKED_ACCOUNT_NBR
	authorization		S	AUTH_CODE
	authReferenceID		S	AUTH_REF_ID
	authReferencePhoneNumber		S	AUTH_REF_PHONE_NBR
	authResponseDetail		S	AUTH_RESP_DETAIL
	authRRN		S	AUTH_RRN
	authRTIEstimatedCPD		S	AUTH_RTI_EST_CPD

	authRTIEstimatedDescription		S	AUTH_RTI_EST_DESCRIPTION
	authRTIEstimatedFee		F	AUTH_RTI_EST_FEE
	authRTIEstimatedProgram		S	AUTH_RTI_EST_PROGRAM
	authSettlementDate		D	AUTH_SETTLE_DT
	authSHA2		S	AUTH_SHA2
	authSource		S	AUTH_SOURCE
	authSpecial1		S	AUTH_SPECIAL_1
	authSpecial2		S	AUTH_SPECIAL_2
	authSpecial3		S	AUTH_SPECIAL_3
	authSpecial4		S	AUTH_SPECIAL_4
	authSpecial5		S	AUTH_SPECIAL_5
	authSpecial6		S	AUTH_SPECIAL_6
	authSpecial7		S	AUTH_SPECIAL_7
	authSpecial8		S	AUTH_SPECIAL_8
	authSpecial9		S	AUTH_SPECIAL_9
	authSpecial10		S	AUTH_SPECIAL_10
	authSpecial11		S	AUTH_SPECIAL_11
	authSpecial12		S	AUTH_SPECIAL_12
	authSpecial13		S	AUTH_SPECIAL_13
	authSpecial14		S	AUTH_SPECIAL_14
	authSpecial15		S	AUTH_SPECIAL_15
	authSpecial16		S	AUTH_SPECIAL_16
	authSpecial17		S	AUTH_SPECIAL_17
	authSpecial18		S	AUTH_SPECIAL_18
	authSpecial19		S	AUTH_SPECIAL_19
	authSpecial20		S	AUTH_SPECIAL_20
	authSTAN		S	AUTH_STAN
	authTermsAcceptance		S	AUTH_TERMS_ACCEPTANCE
	authTLVSets		S	AUTH_TLV_SETS
	authTotalAuthorized		F	AUTH_TOTAL_AUTHORIZED
	authTranDateGMT		D	AUTH_TRAN_DATE_GMT
	authTranId		S	AUTH_TRAN_IDENT
	bankIdReference	^([0-9A-Z]){1,16}$	S	BANK_ID_XREF
	barcodeData	^([0-9A-Fa-f]){4,1024}$	S	BARCODE_DATA
	batchCreditCount		I	BATCH_CREDIT_COUNT
	batchCreditTotal		F	BATCH_CREDIT_TOTAL
	batchDebitCount		I	BATCH_DEBIT_COUNT
	batchDebitTotal		F	BATCH_DEBIT_TOTAL
	batchID	^([0-9]){0,10}$	I	BATCH_ID
	birthDate		D	BIRTH_DATE
	bric	^([A-Z0-9*]){0,20}|BATCH ITEM REV$	S	ORIG_AUTH_GUID
	businessFormatCode		S	BUSINESS_FORMAT_CODE
	cardEntryMethod		S	CARD_ENT_METH
	cardID		S	CARD_ID
	cardmemberReferenceNumber		S	CARDMEMBER_REF_NBR
	cardSequenceNumber		S	CARD_SEQUENCE_NBR
	carrierCode		S	CARRIER_CODE
	carrierName		S	CARRIER_NAME

	cashBackAmount		F	CASH_BK_AMT
	cavvResponse		S	CAVV_RESP
	cavvUCAF		S	CAVV_UCAF
	cellPhone	^.{0,10}$	S	PHONE_CELL
	checkNumber		I	CHECK_NBR
	checkType		S	CHECK_TYPE
	chgType		S	CHG_TYPE
	chipConditionCode		S	CHIP_CONDITION_CODE
	closedDate		S	CLOSED_DATE
	cofPeriod	^([0-9]|0[0-9]|1[0-9]|2[0-4]){0,1}$	I	COF_PERIOD
	convenienceFee	^(\d{1,12}\.)|(\d{0,4}\.\d{1,8})|(\d{0,5}\.\d{1,7})|(\d{0,6}\.\d{1,6})|(\d{0,7}\.\d{1,5})|(\d{0,8}\.\d{1,4})|(\d{0,9}\.\d{1,3})|(\d{0,10}\.\d{1,2})|(\d{0,11}\.\d{1,1})$	F	CONVENIENCE_FEE
	countryCode		S	COUNTRY_CODE
	credentialUsage	^[AIRL]$	S	CREDENTIAL_USAGE
	currencyCode		S	CURRENCY_CODE
	cvv2	^([0-9]){3,4}$	S	CVV2
	directoryServerTranId	^([A-Fa-f0-9\-]){36}$	S	DIRECTORY_SERVER_TRAN_ID
	driversLicenseNumber		S	DL_NBR
	driversLicenseState		S	DL_STATE
	e2ee		S	E2EE
	emailAddress		S	EMAIL
	emvData		S	EMV_DATA
	encAccountNumber		S	ENC_ACCOUNT_NBR
	enhancedTlv		S	ENHANCED_TLV
	expirationDate	^(([0-9][0-9](0[1-9]|1[0-2]))|0000)$	S	EXP_DATE
	extendedPaymentCode		S	EXTENDED_PAYMENT_CODE
	fileCreditCount		I	FILE_CREDIT_COUNT
	fileCreditTotal		F	FILE_CREDIT_TOTAL
	fileDebitCount		I	FILE_DEBIT_COUNT
	fileDebitTotal		F	FILE_DEBIT_TOTAL
	fileID		S	FILE_ID
	folioNumber		S	FOLIO_NBR
	foodAmount		F	FOOD_AMT
	foodID		S	FOOD_ID
	forceClose		S	FORCE_CLOSE
	fromTelephoneNumber		S	FROM_TELEPHONE_NBR
	homePhone	^.{0,10}$	S	PHONE_HM
	hsmIndex		I	HSM_INDEX
	identificationNumber		S	IDENT_NBR
	idExpirationDate		S	ID_EXP_DATE
	idIssuedBy		S	ID_ISSUED_BY
	idNumber		S	ID_NBR
	idType		S	ID_TYPE
	industryType	^([E,M,P]){1}$	S	INDUSTRY_TYPE
	invoiceNumber	^.{0,25}$	S	INVOICE_NBR
	issueCity		S	ISSUE_CITY

	issueNumber		S	ISSUE_NBR
	issueState		S	ISSUE_STATE
	lineNumber		I	LINE_NBR
	localDate		D	LOCAL_DATE
	localTime		S	LOCAL_TIME
	mac		S	MAC
	markedToSettle		S	SETTLE_STATUS
	micrData		S	MICR_DATA
	micrType		S	MICR_TYPE
	msgVersion		S	MSG_VERSION
	multiline		S	MULTILINE
	networkResponse		S	NETWORK_RESPONSE
	orderNumber		S	ORDER_NBR
	orginalTranType		S	ORIG_TRAN_TYPE
	originalAuthAmount	^(([0-9]){1,12}?(\.[0-9]{0,4}) | ([0-9]){0,12}?(\.[0-9]{1,4}))$	F	ORIG_AUTH_AMOUNT
	originalAuthTranId	^([A-Z0-9]){0,20}$	S	ORIG_AUTH_TRAN_IDENT
	originalBatchID	^([0-9]){0,10}$	I	ORIG_BATCH_ID
	originalRequest		S	ORIG_REQUEST
	originalTicketNumber		S	ORIG_TICKET_NBR
	originalTranNumber	^([0-9]){0,10}$	I	ORIG_TRAN_NBR
	payload1		S	PAYLOAD_1
	payload2		S	PAYLOAD_2
	payload3		S	PAYLOAD_3
	payload4		S	PAYLOAD_4
	paymentInitiationChannel	^([0-9]){1,2}$	S	PAYMENT_INITIATION_CHANNEL
	pinBlock		S	PIN_BLK
	pinClear	^([0-9]){4,10}$	I	PIN_CLEAR
	readerStatus		S	READER_STATUS
	reasonCode		S	REASON_CODE
	recvName		S	RECV_NAME
	referenceNumber	^.{0,25}$	S	REFERENCE_NBR
	response		S	AUTH_RESP
	retry	^([1-9]){1}$	S	RETRY
	routingNumber		S	ROUTING_NBR
	salesTax		F	SALES_TAX
	sameDayACH	^([NY]){1}$	S	SAME_DAY_ACH
	scaExemptionIndicator	^([A-F0-9]){4}$	S	SCA_EXEMPTION_INDICATOR
	serverOperator		S	SERVER_OPR
	settlementReversal		S	SETTLEMENT_REVERSAL
	signature		S	SIGNATURE
	signatureFormat		S	SIGNATURE_FORMAT
	softDescriptor		S	SOFT_DESCRIPTOR
	softDescriptor2		S	SOFT_DESCRIPTOR_2
	sourceProtocol		S	SOURCE_PROTOCOL
	ssn		S	SSN
	standardEntryClass		S	STD_ENTRY_CLASS
	startDate		D	START_DATE
	sysPerf		S	SYSPERF

	systemRequest		S	SYSTEM_REQUEST
	tavv		S	TAVV
	tavvECI		S	TAVV_ECI
	taxAmount		F	TAX_AMT
	taxExempt	^([NY]){1}$	S	TAX_EXEMPT
	tdsVersion		S	TDS_VER
	termLocNumber		S	TERM_LOC_NUM
	text		S	AUTH_RESP_TEXT
	ticketID		S	TICKET_ID
	ticketNumber		S	TICKET_NBR
	tidXREF	^([0-9A-Z]){1,16}$	S	TID_XREF
	timeOfCall		S	TIME_OF_CALL
	tipAmount		F	TIP_AMT
	tipAmount1		F	TIP_AMT1
	tipAmount2		F	TIP_AMT2
	tipRecip1		S	TIP_RECIP1
	tipRecip2		S	TIP_RECIP2
	tlvSets		S	TLV_SETS
	tokenTranId		S	TOKEN_TRAN_IDENT
	trace		S	TRACE
	trackData		S	TRACK_DATA
	tran_fee		F	TRAN_FEE
	transaction	^([0-9]){0,10}$	I	TRAN_NBR
	vault1		S	VAULT1
	verboseResponse		S	VERBOSE_RESPONSE
	voiceIVRRequest		S	VOICE_IVR_REQ
	void		S	VOID
	workPhone	^.{0,10}$	S	PHONE_WK