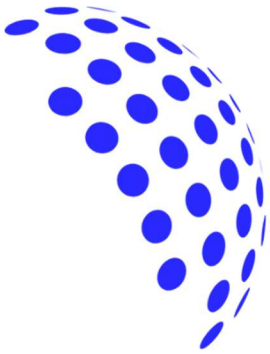


Inovio

Gateway Payment Service API

Merchant Specification

Version 4.14

Table of Contents

1 Overview 11

1.1 Audience 11

1.2 Payment Service 11

1.3 Requirements 11

1.4 Data Security 11

1.5 Revision History 11

2 Authentication 14

2.1 Payment Service Location 14

2.2 Encoding 14

2.3 Service Authentication Test 14

2.3.1 Authentication Fields 14

2.3.2 Authentication Request Example 15

2.3.3 Authentication Response Example 15

2.3.4 Failed Authentication Response Example 16

2.4 Check Service Availability 16

2.4.1 Check Service Availability Fields 16

2.4.2 Check Service Availability Request Example 17

2.4.3 Check Service Availability Response Example 17

2.4.4 Failed Check Service Availability Response Example 17

3 Service Response Format 18

4 Transaction Request Fields 18

4.1 Authentication Parameters 18

4.2 Customer Parameters 19

4.3 Payment and Bank Information Parameters 20

4.4 Line Items Parameters 21

4.5 Adjustment Parameters 21

4.6 Other Merchant Specified Parameters 21

4.7 Membership Adjustment Parameters 24

4.8 Ephemoral Tokenization 25

4.8.1 Requesting the ephemoral token 25

4.8.2 CCAUTHCAP Request Using an Ephemoral Tokenization 27

5 Credit Card Transactions 28
5.1 Authorization 28
5.1.1 Authorization Parameters 28
5.1.2 Authorization Request Example 32
5.1.3 Authorization Response Example 32
5.1.4 Failed Authorization Attempt Response Example 33
5.1.5 Service Declined Response Example 33
5.2 Delayed Capture 34
5.2.1 Delayed Capture Request 34
5.3 Authorization and Capture (Sale) 35
5.3.1 Authorization Request Example (CCAUTHCAP) 35
5.4 Reversal 35
5.4.1 Reversing or voiding an Authorization (CCREVERSE). 35
5.4.2 CREDIT_ON_FAIL Flag 36
5.5 Credit 38
5.5.1 Minimum required parameters for issuing a Credit (CCCREDIT): 38
5.5.2 Force Credit Request 38
5.5.3 Multiple Line Items 38
5.5.4 Example CCAUTHORIZE request with Multiple Line Items. 39
5.5.5 Example CCAUTHORIZE Response with Multiple Line Items 39
5.5.6 Example Request for Capturing a Specific Line Item 40
5.5.7 Capturing, Reversing or Crediting the Entire Order with Multiple Line Items 41
5.5.8 Example Request for Sending Reversal of a Specific Line Item 42
5.5.9 Line Item Types 44
5.6 Checking the Status of an order 45
5.6.1 CCSTATUS Example Request using REQUEST_REF_PO_ID 45
5.6.2 CCSTATUS Example Request using REQUEST_REF_PO_ID_XTL 45
5.6.3 CCSTATUS Example Response 45
5.6.4 CCSTATUS Response Fields 47
5.6.5 Transaction Status is PENDING 47
5.7 Updating Order Data 48
5.7.1 CCTRANSUPDATE Example Request using REQUEST_REF_PO_ID 48

5.7.2 CCTRANSUPDATE Example Request using REQUEST_REF_PO_ID_XTL 48
5.7.3 CCTRANSUPDATE Example Response 48
5.7.4 CCTRANSUPDATE Response Fields 49
6 ACH/eCheck Transactions 49
6.1 Transaction State Updates Through Postback (Optional) 50
6.2 Transaction State using Order Detail API (Optional) 50
6.2.1 Transaction Status 50
6.3 Gateway Request Parameters 50
6.4 Gateway Actions 51
6.5 Authorization and Capture (ACHAUTHCAP) Request and Response 51
6.5.1 Authorization (ACHAUTHCAP) Request Example 51
6.5.2 Authorization (ACHAUTHCAP) Response Example (Pending Status) 52
6.6 Account Validation via AUTHORIZATION without CAPTURE (ACHAUTHORIZE) Request and Response 52
6.6.1 Authorization (ACHAUTHORIZE) Request Example 53
6.6.2 Authorization (ACHAUTHORIZE) Response Example (Pending Status) 53
6.6.3 Service Declined Response Example 54
6.7 Reversal Request and Response 54
6.7.1 Reversal Request Example 54
6.7.2 Reversal Response Example 54
6.8 Credit Request and Response 55
6.8.1 Credit Request 55
6.8.2 Credit Response 55
6.9 Status Change 55
7 Boleto 55
7.1 Request Parameters 56
7.2 Additional Response Parameters 57
7.2.1 Boleto Request Example 57
7.2.2 Boleto Response Example 57
8 Pix Payment 58
8.1 PIX Request Parameters 58
8.2 Additional Response Parameters 59

8.3 Pix Request Example 59
8.4 Pix Response Example 59
9 Credilink 60
9.1 How it Works 60
9.2 How to Use Credilink 61
9.3 Additional Parameters 61
10 Order Insight 62
10.1 Additional Request Parameters 62
10.2 Reporting 63
10.2.1 Order Insight Lookup Statuses 63
11 Transaction Modes 63
11.1 Recurring 63
11.1.1 Minimum required fields for sending a simple recurring transaction request using CUST_ID: 64
11.1.2 Required fields for specifying which payment account to charge with the recurring transaction request using PMT_L4 parameter: 64
11.1.3 Required fields for specifying which payment account to charge with the recurring transaction request using PMT_ID parameter: 65
11.1.4 Required fields for specifying which payment account to charge with the recurring transaction request using PMT_ID_XTL parameter: 65
11.1.5 Required parameters for sending new billing information with the recurring transaction request: 66
11.1.6 Parameters for sending Renewal Transactions (Rebills): 67
11.2 Memberships 67
11.2.1 Suggested fields for sending a membership transaction (new customer): 68
11.2.2 Parameters to create Subscription directly from the Gateway API 69
11.2.3 Unique Customer Login Check 70
11.2.4 Membership Cancelation Requests (SUB_CANCEL) 71
11.2.5 Membership Product Update Requests (Upgrade/downgrade Product) 73
12 Response Fields 75
12.1 Response Formats 75
12.2 Successful Transaction Response Fields 75
12.3 Declined Transaction Response Fields 77

12.4 Service Declined Response Fields 78
13 Dynamic Descriptor 79
13.1 Dynamic Descriptor 79
13.1.1 Dynamic Descriptor Restrictions 79
14 Transaction Validation and Risk Assessment 79
14.1 Address Verification Service 79
14.1.1 Check AVS Flag (CHKAVS) 80
14.1.2 Positive AVS Response Codes 80
14.1.3 Negative AVS Response Codes 81
14.1.4 Custom Settings (AVSMatchSet) 82
14.2 Card Security Code 82
14.2.1 Check CVV Flag (CHKCVV) 82
14.2.2 Positive CVV Response Codes 83
14.2.3 Negative CVV Response Codes 83
14.2.4 Custom Settings (CVVMatchSet) 83
14.3 Timeout Void 84
14.3.1 Timeout Void Parameters 84
14.3.2 Timeout Void Portal Setting 84
15 3D Secure Transactions 84
15.1 3D Secure 84
15.1.1 Low Value Strong Customer Authentication (SCA) Exemption 85
15.1.2 External 3DS Provider Scenario 85
15.1.3 Payment Gateway as 3DS Provider Using Redirect Scenario 86
15.1.4 Determining which outcome happened 89
15.1.5 Special cases 92
15.1.6 3DS Browser & Device Data 95
16 Currency 100
16.1 Multi-currency Support 100
16.2 Request_currency Parameter 101
16.3 Currency Response Fields 101
16.3.1 Testing Multi-currency Response 101
16.3.2 Multi-Currency Response Example 101

17 Partial Authorization 102
17.1 What is Partial Authorization? 102
17.1.1 How It Works 102
17.1.2 Partial Authorization Required Parameters 102
17.1.3 Failed Transaction Due to Minimum Amount Response Example 103
17.1.4 Partial Authorization Successful Response Example 103
18 External Order ID Uniqueness 104
18.1 UNIQUE_XTL_ORDER_ID Parameter 104
18.1.1 UNIQUE_XTL_ORDER_ID Setting 104
18.1.2 Example Decline Response when UNIQUE_XTL_ORDER_ID parameter is set to “1”: 105
19 Processor Specific Fields 105
19.1 Processor Specific Fields 105
20 PagoEfectivo 106
20.1 PagoEfectivo Features 106
20.2 PagoEfectivo Request Parameters 106
20.2.1 userDocumentType Explained 107
20.3 Additional Response Parameters 107
20.4 PagoEfectivo Request Example 108
20.5 PagoEfectivo Response Example 108
21 Apple Pay 109
21.1 Overview 109
21.2 Customers who can use Apple Pay 109
21.3 Flow 109
21.4 Fetch Apple Pay configuration data from the gateway 110
21.5 Update the configuration with the specific payment/order details 112
21.6 Add the Apple Pay library to your payment page 112
21.7 Create a JS function to validate your domain 113
21.8 Present the Apple Pay button 113
21.9 Create and start the Apple Pay session 114
21.9.1 Handle the Authorized Payment 114
21.9.2 Handle a Canceled Payment or Error 114
21.10 Send the authorized payment data from the session to the gateway 115

21.11 Important requirements for Apple Pay authorization requests 115
21.12 Rebiling and Credentials on File 116
22 Google Pay 116
22.1 Overview 116
22.2 Implementation Process Flow 116
22.3 Fetch your Google Pay configuration data from the gateway 116
22.4 Add the Google Pay library to your payment page 117
22.5 Customizing the Google Pay button 118
22.6 Handling the Google Pay Loaded Event 118
22.7 Configuring Google Pay 119
22.8 Handling the Google Pay Button Click 120
22.9 Payment Authorization 125
22.10 Submitting authorized payment data 126
22.10.1 Important requirements for Google Pay authorization requests 126
22.11 Rebill and Credentials on File 126
23 Scheme Level Tokenization 126
23.1 Additional Response Parameter 127
24 Service Provider Tokens: Merchant Decryption 127
24.1 Overview 127
24.2 Request Parameters 127
25 Single Euro Payments Area (SEPA) 128
25.1 SEPA Direct Debit Payment Process 128
25.2 Transaction state updates through Postback 128
25.3 Authorization Request (SEPA Direct Debit) - DBTAUTHORIZE 129
25.4 Authorization Capture Request (SEPA Direct Debit) - DBTCAPTURE 129
25.5 Debit Request Parameters (DBTAUTHORIZE) 129
25.5.1 Debit Request Example 130
25.5.2 Debit Response Example (Pending) 130
25.5.3 Debit Response Example (Success) 131
25.6 Reversals 132
25.6.1 Reversing a mandate (DBTREVERSE) 132
25.6.2 DBTREVERSE transaction Request Example 132

25.7 Refunds or Credits 133
25.7.1 Issuing refunds or credits (DBTCREDIT) 133
25.8 Recurring transactions 133
25.8.1 Recurring transaction Request Example (DBTDEBIT & request_rebill=2) 133
25.8.2 Recurring transaction Response Example (DBTDEBIT & request_rebill=2) 133
25.8.3 Recurring transaction Request Example (DBTDEBIT & request_rebill=1) 134
25.8.4 Recurring transaction Response Example (DBTDEBIT & request_rebill=1) 135
25.9 Additional Parameters 136
26 iDEAL Payments 136
26.1 Debit Request Parameters (DBTAUTHORIZE) 136
26.1.1 Debit Request Example 138
26.1.2 Debit Response Example (Pending) 138
26.1.1 Debit Response Example (Success) 139
26.2 Reversals 140
26.2.1 Reversing a mandate (DBTREVERSE) 140
26.2.2 DBTREVERSE transaction Request Example 140
26.3 Refunds or Credits 140
26.3.1 Issuing refunds or credits (DBTCREDIT) 140
26.4 Recurring transactions 141
26.4.1 Recurring transaction Request Example (DBTDEBIT & request_rebill=2) 141
26.4.2 Recurring transaction Response Example (DBTDEBIT & request_rebill=2) 141
26.4.3 Recurring transaction Request Example (DBTDEBIT & request_rebill=1) 142
26.4.4 Recurring transaction Response Example (DBTDEBIT & request_rebill=1) 143
26.5 Additional Parameters 144
27 EPS Payments 144
27.1 Debit Request Parameters (DBTAUTHORIZE) 144
27.1.1 Debit Request Example 146
27.1.2 Debit Response Example (Pending) 146
27.1.1 Debit Response Example (Success) 147
27.2 Reversals 148
27.2.1 Reversing a mandate (DBTREVERSE) 148
27.2.2 DBTREVERSE transaction Request Example 148

27.3 Refunds or Credits 149
27.3.1 Issuing refunds or credits (DBTCREDIT) 149
27.4 Recurring transactions 149
27.4.1 Recurring transaction Request Example (DBTDEBIT & request_rebill=2) 149
27.4.2 Recurring transaction Response Example (DBTDEBIT & request_rebill=2) 149
27.4.3 Recurring transaction Request Example (DBTDEBIT & request_rebill=1) 150
27.4.4 Recurring transaction Response Example (DBTDEBIT & request_rebill=1) 151
27.5 Additional Parameters 152
28 Testing 152
28.1 Using the test bank 152
28.2 Basic Gateway Test 152
28.3 Test Credit Cards 154
Appendix A: Service Request Types 155
Appendix B: Transaction Status 156
Appendix C: API Response Codes 157
Appendix D: Service Response Codes 160
Appendix E: Address Verification Service Codes 164
Appendix F: CVV Response Codes 165
Appendix G: Negative Option Billing- MCC 5968 166
Appendix H: 3-D Secure Flow 168
Appendix I: Automatic Account Updater (AAU) 169
Appendix J: Visa Trial Processing 169
Appendix K: Card on File Matrix 170

1 Overview

1.1 Audience

This documentation is intended to be used by the merchant organization's technical staff. This document assumes that the reader has a working knowledge of programming languages, hardware and software requirements specified in this documentation.

1.2 Payment Service

The Payment Service is designed to allow merchants communicate and process online transactions with the payment gateway's transaction processing system. The API serves as a bridge between the merchant's website and various financial institutions in processing real-time payment transactions.

1.3 Requirements

To integrate with the Payment Service API, the following are required:

- Internet Connection
- Knowledge of HTTP Post and Get methods
- Knowledge of parsing data in any of the following formats: XML and JSON

1.4 Data Security

For security and guarding confidentiality, all cardholder and transaction data are sent over the internet using the TLS 1.2 cryptographic protocol. All Cardholder data is encrypted at all times. The payment gateway is **PCI-DSS Level 1** compliant.

1.5 Revision History

Version	Date	Description
2.0	3 Jul 2012	Added parameters for membership transactions. Updated requirement for REQUEST_API_VERSION parameter. Added Membership in Transaction Modes (Section 6). Added SUB_CANCEL and SUB_UPDATE request action types. Updated Response Fields (Section 7). Updated all gateway request and response examples. Added Cross-sale transactions descriptions. Updated Service Response table in Appendix section. Removed REQUEST_TYPE and REQUEST_SCRUB_FLAG fields from Transaction Request Fields.
2.1	18 Jul 2012	Added REQUEST_AFF_ID_SUB to the list of gateway fields. Update REQUEST_CURRENCY description. Updated BILL_ADDR_STATE description.
2.2	18 Sept 2012	Added MERCH_ACCT_ID as a required parameter on recurring transactions (Section 6).
3.2	21 Jan 2014	Added card type fields in gateway response (Section 8). Added Mastercard EMS Fraud Score parameters (new section, 16). Formatting changes (all sections).
3.3	28 Mar 2014	Updated Mastercard EMS Fraud Score section with gateway response fields. Added gateway response examples and EMS Reason Codes table.

3.4	17 Sep 2014	Added PROC_UDF fields in Transaction Request Parameters table. Added new section for merchants using eMerchantPay banks and Threatmetrix™.
3.5	9 Oct 2014	Added format for CUST_BIRTHDAY field. Updated Transaction Request and Response Parameters tables (Sections 4 and 8). Updated 3D Secure section. Updated Appendix C: Transaction Status. Added "Pending" status explanation (Section 5.6.5). Updated Section 17: Processor Specific Fields (this was changed from being a Threatmetrix-specific section).
3.6	14 May 2015	Added CARD_BALANCE in Response Fields (Section 8.2) and gateway response examples. Added REQUEST_CURRENCY field on recurring transaction request parameters (Section 7). Added Partial Auth testing using test banks (Section 18).
3.7	12 Feb 2016	Removed Deprecated Payment Types Appendix Extension of UDF up to XTL_UDF20
3.8	6 Oct 2016	Added CUST_BRCPFCNPJ parameter to send CPF or CNPJ with Brazilian transactions
3.9	14 Mar 2017	Added PMT_DESCRIPTOR_PHONE for dynamic support phone number (aka CITY field) on customer descriptor. Reference section 9.1
3.9.1	18 Aug 2017	Extended FORCE_CREDIT ability to support OCT
3.9.2	5 Sept 2017	Description of tokenization feature and its use
4.0	1 June 2018	Description of process and parameters for 3-D Secure transactions
4.1	9 July 2018	Added test scenario information for multi-currency processing , and additional response assertion test scenarios
4.2	27 Feb 2019	Added Support for ACHAUTHORIZE for ACH authorization without transferring any funds. Also added CCTRANSUPDATE for updating orders with Receipts, Added TRANS_REBILL_TYPE, MBSHP_ID_XTL, TRANS_TRIAL_REBILL_CUSTOMER_CONSENT, TRANS_CUSTOMER_RECEIPT Request Parameters to store MCC5968-Mastercard rule, added CARD_ON_FILE_FLAG parameter
4.3	12 Jul 2019	Added Automatic Account Updater (AAU) Update Date; PMT_AAU_UPDATE_DT and Update type Description; PMT_AAU_UPDATE_DESC. Also added Appendix I for AAU Services
4.4	13 Jul 2020	Added COF parameters including request_initiator, request_installment, pmt_numb_cof and added option "2" on request_rebill parameter. Also Added Appendix J for Visa Trial Processing and revised TRANS_REBILL_TYPE, MBSHP_ID_XTL, TRANS_CUSTOMER_RECEIPT Request Parameters to store VISA Trial processing. Also added Service Response Code '519'for 'Missing Trial Descriptor'. Added two additional parameters for 3DS (P3DS_VERSION P3DS_TRANSID). Also Added CARD_DETAIL (Credit/Debit) Field in our Response. Removed support for Double Pipes

		Delimited. Removed Card_on_file_flag field (deprecated by request_initiator). Removed SEPA Section (not supported)
4.5	20 April 2022	Added new Service Response Codes, added Appendix J: Card on File Matrix, Added Parameters to create Subscription directly from Gateway API, Added Boleto Payment Processing, Added Pix Payment Processing, Added Brazilian Credilink, added SERVICE_RESPONSE, SERVICE_ADVICE, PROCESSOR_RESPONSE, PROCESSOR_ADVICE to CCSTATUS, Added REQ_ID in all Responses, Added PMT_NUM to scrub decline response, Added 3DS 2.0 Guide
4.6	30 August 2023	Removed FORCE_NEW_CUST from PIX Request, Added REQ_LOW_VALUE_SCAEXEMPTION Field, Added a new REQUEST_ACTION - PAGSALE for PagoEfectivo, Added Digital Wallet Parameters PMT_WALLET AND PMT_CRYPTOGRAM, Added Cash Discount's CONVENIENCE_FEE Field, Added TAX_EXEMPT and TAX_AMT for Level II Processing, updated Appendix D -Service Response Codes with new Scrub decline response codes, added CCRDR Request Action, Deprecated Outcome 2: 3DS version 1 authentication, Added PagoEfectivo Payment Section, Added Apple Pay Section
4.7	03 January 2024	Added Scheme Level Tokenization Section, Added Scheme Level Tokenization Parameters TRANS_NTOKEN_USED, ORIG_CARD_BRAND_TRANSID and CARD_BRAND_TRANSID, Added New Service Response Codes (700, 720, 725), updated Credilink Section with current functionalities
4.8	06 March 2024	Added Single Euro Payments Area (SEPA) Direct Debit Support, added new Request Actions for Debit Payment Type Support; DBTAUTHORIZE, DBTCAPTURE DBTDEBIT, DBTREVERSE and DBTCREDIT, added DEBIT_TYPE parameter to support Single Euro Payments Area (SEPA), iDEAL Payments and EPS Payments. Revised CCAUTHORIZE and CCCAUTHCAP gateway timeout from 90 Secs to 120 Secs. Also added ACHPAYOUT used for ACH Direct PAYOUT, corrected Apple Pay FetchUrl from https://api.inoviopay.com/apple-pay-poc/api/session/create to https://api.inoviopay.com/apple-pay-services/api/session/create, Added SUB_UPDATE_PMT_ID Req/Resp examples
4.9	02 February 2025	Added Tokenization updates, additional Response Paramaters, Service Provider Tokens: Merchant Decryption, Verifi: Order Insight, CCRDRDELETE to Service Request Types, new API Response Codes, and general document cleanup
4.10	05 May 2025	Added Network Tokens to the Service Provider Tokens: Merchant Decryption section
4.11	03 July 2025	Added Google Pay and Google Pay Response Codes
4.12	16 September 2025	Added PMT_DESCRIPTOR_CITY request parameter, 3DS Endpoint Update, and general document cleanup

4.13	16 December 2025	Updated SUB_CANCEL Gateway Response Example, Added Timeout Void setting, Added PagoEfectivo userDocumentType details, Added SUB_UPDATE_TYPE to Membership Updates
4.14	06 March 2026	Added 3DS Browser & Device Data Fields Reference

2 Authentication

2.1 Payment Service Location

The Payment Service is located at:
https://api.inoviopay.com/payment/pmt_service.cfm

2.2 Encoding

The API expects all data to be URL and UTF 8 encoded, using character set, iso-8859-1 over HTTPS.

Value	Encoded Value	Description
&	Not needed	Used to separate all parameters
=	Not needed	Used for all parameter values
|	Not needed	Used to separate all parameters within the REQUEST_FILTER variable
+	%2B	Used for all parameter values within the REQUEST_FILTER variable
SPACE	%20	If a space is needed, be sure to encode the character

2.3 Service Authentication Test

The first requirement in integrating with the Payment Service is to have a working Service login name (REQ_USERNAME), password (REQ_PASSWORD), and website ID (SITE_ID).

2.3.1 Authentication Fields

Field Name	Description
REQ_USERNAME	Merchant's Service username
REQ_PASSWORD	Merchant's Service password
SITE_ID	Merchant's website ID
REQUEST_ACTION	Service request action: **TESTAUTH**
REQUEST_RESPONSE_FORMAT	Service response format.

REQUEST_API_VERSION	API Version of this document. (4.14) Must be sent on all gateway requests.

2.3.2 Authentication Request Example

Below is a complete HTTP Post request example (including the header information) that contains the required Fields needed when testing basic gateway authentication:

```
POST /payment/pmt_service.cfm HTTP/1.0
User-Agent: Mozilla/ 117.0.1 (compatible; MS Edge 117.0.2045.36; Windows NT 5.1)
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash,
application/vnd.ms-powerpoint, application/vnd.ms-excel, application/msword, */*
Accept-Encoding: identity
Accept-Language: en-us,en
Host: my.host.invalid
Accept-Charset: iso-8859-1,*,utf-8
Content-Type: application/x-www-form-urlencoded
Content-Length: 137
req_username=test@example.com&req_password=P5ssw0rd!1&request_action=TESTAUTH&site
_id=0&request_response_format=XML&request_api_version=4.14
```

2.3.3 Authentication Response Example

Each Service request will produce a corresponding response document

```
<?xml version="4.14" encoding="UTF-8"?>
<RESPONSE>
<REQUEST_ACTION/>
<TRANS_STATUS_NAME/>
<TRANS_VALUE/>
<TRANS_ID/>
<CUST_ID/>
<XTL_CUST_ID/>
<MERCH_ACCT_ID/>
<CARD_BRAND_NAME/>
<PMT_L4/>
<API_RESPONSE>0</API_RESPONSE>
<API_ADVICE> </API_ADVICE>
<SERVICE_RESPONSE>100</SERVICE_RESPONSE>
<SERVICE_ADVICE>User Authorized</SERVICE_ADVICE>
<PROCESSOR_RESPONSE>0</PROCESSOR_RESPONSE>
<PROCESSOR_ADVICE> </PROCESSOR_ADVICE>
<INDUSTRY_RESPONSE>0</INDUSTRY_RESPONSE>
<INDUSTRY_ADVICE> </INDUSTRY_ADVICE>
<REF_FIELD/>
<PROC_ID/>
<PROC_NAME/>
<AVS_RESPONSE/>
<CVV_RESPONSE/>
<REQUEST_API_VERSION>4.14</REQUEST_API_VERSION>
</RESPONSE>
```

2.3.4 Failed Authentication Response Example

Below is an example response of a failed authentication request. API_RESPONSE “101” indicates a failed authentication attempt.

```
<?xml version="1.0" encoding="UTF-8"?>
<RESPONSE>
<REQUEST_ACTION/>
<TRANS_STATUS_NAME/>
<TRANS_VALUE/>
<TRANS_ID/>
<CUST_ID/>
<XTL_CUST_ID/>
<MERCH_ACCT_ID/>
<CARD_BRAND_NAME/>
<PMT_L4/>
<PMT_ID/> 441402</PMT_ID>
<PMT_ID_XTL/>
<API_RESPONSE>101</API_RESPONSE>
<API_ADVICE>Invalid login information</API_ADVICE>
<SERVICE_RESPONSE>0</SERVICE_RESPONSE>
<SERVICE_ADVICE>Declined</SERVICE_ADVICE>
<PROCESSOR_RESPONSE>0</PROCESSOR_RESPONSE>
<PROCESSOR_ADVICE> </PROCESSOR_ADVICE>
<INDUSTRY_RESPONSE>0</INDUSTRY_RESPONSE>
<INDUSTRY_ADVICE> </INDUSTRY_ADVICE>
<REF_FIELD/>
<PROC_NAME/>
<AVS_RESPONSE/>
<CVV_RESPONSE/>
<REQUEST_API_VERSION>4.14</REQUEST_API_VERSION>
</RESPONSE>
```

2.4 Check Service Availability

Use request action, “TESTGW”, to check if Payment Service is available to process requests.

2.4.1 Check Service Availability Fields

Field Name	Description
REQ_USERNAME	Merchant’s Service username
REQ_PASSWORD	Merchant’s Service password
REQUEST_ACTION	Service request action: **TESTGW**
SITE_ID	Merchant’s website ID
REQUEST_RESPONSE_FORMAT	Service response format
REQUEST_API_VERSION	API Version (4.14)

2.4.2 Check Service Availability Request Example

```
POST /payment/pmt_service.cfm HTTP/1.0
User-Agent: Mozilla/4.6 (compatible; MSIE 7.0; Windows NT 5.1)
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-
flash, application/vnd.ms-powerpoint, application/vnd.ms-excel, application/msword,
*/*
Accept-Encoding: identity
Accept-Language: en-us,en
Host: Host.example.com
Accept-Charset: iso-8859-1,*,utf-8
Content-Type: application/x-www-form-urlencoded
Content-Length: 112
req_username=test@example.com&req_password=P5ssw0rd!1&request_action=TESTGW&site_id=0
&request_response_format=XML&request_api_version=4.14
```

2.4.3 Check Service Availability Response Example

```
<?xml version="1.0" encoding="UTF-8"?>
<RESPONSE>
<REQUEST_ACTION/>
<TRANS_STATUS_NAME/>
<TRANS_VALUE/>
<TRANS_ID/>
<CUST_ID/>
<XTL_CUST_ID/>7
<MERCH_ACCT_ID/>
<CARD_BRAND_NAME/>
<PMT_L4/>
<PMT_ID/>
<PMT_ID_XTL/>
<API_RESPONSE>0</API_RESPONSE>
<API_ADVICE> </API_ADVICE>
<SERVICE_RESPONSE>101</SERVICE_RESPONSE>
<SERVICE_ADVICE>Service Available</SERVICE_ADVICE>
<PROCESSOR_RESPONSE>0</PROCESSOR_RESPONSE>
<PROCESSOR_ADVICE> </PROCESSOR_ADVICE>
<INDUSTRY_RESPONSE>0</INDUSTRY_RESPONSE>
<INDUSTRY_ADVICE> </INDUSTRY_ADVICE>
<REF_FIELD/>
<PROC_ID/>
<PROC_NAME/>
<AVS_RESPONSE/>
<CVV_RESPONSE/>
<REQUEST_API_VERSION>4.14</REQUEST_API_VERSION>
</RESPONSE>
```

2.4.4 Failed Check Service Availability Response Example

```
<?xml version="1.0" encoding="UTF-8"?>
<RESPONSE>
<REQUEST_ACTION/>
<TRANS_STATUS_NAME/>
<TRANS_VALUE/>
<TRANS_ID/>
```

```
<CUST_ID/>
<XTL_CUST_ID/>
<MERCH_ACCT_ID/>
<CARD_BRAND_NAME/>
<PMT_L4/>
<PMT_ID/>
<PMT_ID_XTL/>
<API_RESPONSE></API_RESPONSE>
<API_ADVICE> </API_ADVICE>
<SERVICE_RESPONSE></SERVICE_RESPONSE>
<SERVICE_ADVICE> Service Unavailable </SERVICE_ADVICE>
<PROCESSOR_RESPONSE>0</PROCESSOR_RESPONSE>
<PROCESSOR_ADVICE> </PROCESSOR_ADVICE>
<INDUSTRY_RESPONSE>0</INDUSTRY_RESPONSE>
<INDUSTRY_ADVICE> </INDUSTRY_ADVICE>
<REF_FIELD/>
<PROC_NAME/>
<AVS_RESPONSE/>
<CVV_RESPONSE/>
<REQUEST_API_VERSION>4.14</REQUEST_API_VERSION>
</RESPONSE>
```

3 Service Response Format

For every request, the Payment Service API returns a response message. Merchants can choose one from the following supported formats: **XML** and **JSON**. To set the Service Response format, send the REQUEST_RESPONSE_FORMAT parameter with your transaction request. In this document, the response examples are in XML format.

4 Transaction Request Fields

A transaction request includes:

- Authentication Parameters
- Customer Parameters
- Transaction and Payment Information Parameters
- Line Item Parameters
- Adjustment Parameters (used for issuing credits, reversals and delayed captures).
- Merchant Specified Parameters

4.1 Authentication Parameters

Field Name	Description
REQ_USERNAME	Merchant's Service Username.
REQ_PASSWORD	Merchant's Service Password.
REQUEST_ACTION	Service Request Action

REQUEST_API_VERSION	Payment Service API Version (4.14) Must be sent on all gateway requests.
REQUEST_RESPONSE_FORMAT	Service response format
SITE_ID	Merchant's Website ID

4.2 Customer Parameters

Field Name	Description
BILL_ADDR	Cardholder Billing Street Address
BILL_ADDR_CITY	Cardholder Billing City
BILL_ADDR_COUNTRY	Cardholder Billing Country
BILL_ADDR_STATE	Cardholder Billing State
BILL_ADDR_ZIP	Cardholder Billing Postal/ZIP code
CUST_BIRTHDAY	Cardholder's date of birth. FORMAT: MM-DD-YYYY
CUST_DLN	Cardholder Driver's License ID
CUST_DLN_STATE	Cardholder Driver's License State (US Customers Only)
CUST_EMAIL	Cardholder Email Address Note: Required by some banks.
CUST_FNAME	Cardholder's First Name Note: Required by some banks.
CUST_LNAME	Cardholder's Last Name Note: Required by some banks.
CUST_LOGIN	Cardholder's Login or User Name
CUST_PASSWORD	Cardholder's Password Password must be at least 10 characters with 1 number, lower case and upper case letter.
CUST_PHONE	Cardholder's Phone Number example *+1 (123) 456–7890* Note: Required by some banks.
CUST_SSN_L4	Cardholder's Last 4 digits of Social Security Number (US Customers Only)
SHIP_ADDR	Cardholder's Shipping Street Address

SHIP_ADDR_CITY	Cardholder's Shipping City
SHIP_ADDR_COUNTRY	Cardholder's Shipping Country
SHIP_ADDR_STATE	Cardholder's Shipping State
SHIP_ADDR_ZIP	Cardholder's Shipping Postal/ZIP Code
SITE_ID	Merchant's Website ID
XTL_IP	Cardholder's IP Address
MBSHP_ID_XTL	External Membership ID
USER_AGENT_XTL	Software agent responsible for retrieving and facilitating end-user interaction with Web content

4.3 Payment and Bank Information Parameters

Field Name	Description
PMT_DESCRIPTOR	Bank Dynamic Descriptor
PMT_DESCRIPTOR_PHONE	Bank Dynamic Customer Support Phone Number (aka City Field)
PMT_DESCRIPTOR_CITY	Bank Dynamic Customer Support City Applicable to MasterCard only
PMT_EXPIRY	Credit Card Expiration Date
PMT_KEY	Credit Card CVV2 or CVC2 Code
PMT_NUMB	Credit Card Number
TOKEN_GUID	Token ID used in place of pmt_numb. See Ephermoral Tokenization.
PMT_L4	Last 4 digits of the account or credit card number
PMT_ID	Payment Unique Identifier
PMT_ID_XTL	External Payment ID
REQUEST_CURRENCY	Currency 3-letter Code See Currency section for more information.
MERCH_ACCT_ID	Merchant Account ID
DEBIT_TYPE	Debit Type used for the transaction. (SEPA, iDEAL, and EPS)

4.4 Line Items Parameters

Field Name	Description
LI_COUNT_1	Line Item Count. Use this parameter to indicate quantity. Max. Value is "10".
LI_PROD_ID_1	Line Item Product ID 1
LI_VALUE_1	Line Item Transaction Amount 1

4.5 Adjustment Parameters

Field Name	Description
REQUEST_REF_PO_ID	Reference Order ID is used when sending adjustment request against authorizations (i.e. Delayed Capture, Reversal and Credit requests).
REQUEST_REF_PO_LI_ID	Reference Line Item ID. Use this if multiple line items processing is needed.
CUST_ID	Customer ID created by the system after sending a successful authorization request.
REQUEST_REF_PO_ID_XTL	Request Reference XTL_ORDER_ID (Merchant's Order ID). Used for sending CCSTATUS requests.
CREDIT_ON_FAIL	Used with reversal (void) requests. If set to 1, the system will automatically credit the transaction when the reversal request failed. See CREDIT_ON FAIL explanation for details.
FORCE_CREDIT	Used for sending Force Credit transactions.

4.6 Other Merchant Specified Parameters

Field Name	Description
CHKAVS	Address Verification Service Flag (this is for enabling, ignoring or disabling the AVS option). **AVS Check is enabled by default.**
AVSMATCHSET	Used for setting the response codes to check when approving transactions based on the AVS response code. See AVSMatchSet explanation for more details.
CHKCVV	CVV Check flag (this is for enabling, ignoring, or disabling the Credit Card CVV2 or CVC2 check). **CVV check is enabled by default. Some processors may automatically decline transactions that return a negative CVV response code.**
CVVMATCHSET	Used for setting the response codes to check when approving transactions based on the CVV response code. See CVVMatchSet explanation for more details.
CUST_BRCPFCNPJ	Individual CPF/Business CPNJ Number – Specific to Brazil

	• See Section 8 for use with Pix Payment and Section 9 for use with Credilink
CONVENIENCE_FEE	Reference Field used to indicate the surcharge fee amount which is included in the amount of the transaction
REQUEST_AFF_ID	External Affiliate ID
REQUEST_API_VERSION	Payment Service API Version (4.14)
REQUEST_LANGUAGE	Language 3-letter Code
REQUEST_RESPONSE_FORMAT	Service Response Format
XTL_UDFXX	Merchant's User Defined Field, with **xx** having a value from 01 to 20, i.e. xtl_udf01, xtl_udf02, xtl_udf03
XTL_ORDER_ID	Merchant's Order ID
XTL_CUST_ID	Merchant's Customer ID
LI_XTL_PROD_ID_X	Merchant's Product ID Name. "x" indicates a dynamic number depending on the line items sent in the request. For more information see the Multiple Line Items section.
REQUEST_INITIATOR	Used in Card on File (COF) transactions. Set this parameter to 'C' for Customer Initiated transactions (CIT). Set this parameter to 'M' for Merchant Initiated transactions (MIT).
REQUEST_INSTALLMENT	Default: "0" Set this parameter to '1" if the transaction is an installment payment.
PMT_NUMB_COF	Default: "0" Set this parameter to "1" to specify that a stored payment number has been used by the merchant's own system for a COF transaction.
REQUEST_REBILL	Default: "0" Set this parameter to "1" to specify that the transaction is a Rebill in a subscription Set this parameter to "2" to specify that the transaction is the FIRST transaction in a subscription
REQUEST_XSALE	Request Cross-sale transaction.
PROC_GUID	Processor GUID/GUWID
P3DS_RETURN_URL	3-D Secure Return URL after verification.
P3DS_PARAMS	3-D Secure Verification Parameters (Payspace accounts with 3DS only) **Note:** Data returned in this field is URL-encoded.
P3DS_VERIFICATION_URL	3-DS Verification URL where customers should be redirected to for external verification.

REQUEST_ENROLLMENT	Transforms a request into a 3-D Secure enrollment check request
P3DS_TRANSID	Transaction ID specific to a 3-D Secure vendor
P3DS_VERSION	3DS Version used on transaction (2)
P3DS_ECI	3-D Secure ECI value
P3DS_SCREEN_HEIGHT	Total height of the cardholder's screen in pixels
P3DS_SCREEN_WIDTH	Total width of the cardholder's screen in pixels
P3DS_JAVA_ENABLED	A Boolean value (TRUE/FALSE) that represents the ability of the cardholder browser to execute Java
P3DS_JAVASCRIPT_ENABLED	A Boolean value (TRUE/FALSE) that represents represents the ability of the cardholder browser to execute JavaScript
P3DS_BROWSER_HEADER	The exact content of the HTTP accept headers sent from the cardholder's browser. Example: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
P3DS_BROWSER_LANGUAGE	Value represents the browser language as defined in IETF BCP47
P3DS_BROWSER_COLOR_DEPTH	Value represents the bit depth of the color palette for displaying images, in bits per pixel Possible Values: 1, 4, 8, 15, 16, 24, 32, 48
P3DS_BROWSER_TIME_ZONE	Time difference between UTC time and the cardholder browser local time, in minutes Note: Regardless of direction value should be positive.
P3DS_CHALLENGE_WINDOW	An override field that a merchant can pass in to set the challenge window size to display to the end cardholder. The ACS will reply with content that is formatted appropriately to this window size to allow for the best user experience. The sizes are width x height in pixels of the window displayed in the cardholder browser window. Possible values: 01 - 250x400, 02 - 390x400, 03 - 500x600, 04 - 600x400, 05 - Full page
REQUEST_PARES	Submitted in a 3-D Secure transaction, a value obtained from an enrollment request
REQUEST_AFF_ID_SUB	External Sub-affiliate ID
REQ_LOW_VALUE_SCAEXEMPTION	Used by Merchants to bypass 3DS on low value transactions by requesting Low-Value SCA Exemption
UNIQUE_XTL_ORDER_ID	Enables External Order ID uniqueness.
PMT_ID_XTL	External Payment /Credit Card Unique ID
PROC_UDF01	Processor Defined Field 1
PROC_UDF02	Processor Defined Field 2

PROC_SUCCESS_URL	Redirect URL after successful 3-D Secure authentication process.
PROC_ERROR_URL	Redirect URL after unsuccessful 3-D Secure authentication process.
PROC_REDIRECT_URL	3DS Authentication URL where customers should be redirected to for authentication.
PROD_NAME	Product Name
PROD_TYPE	1- Membership Cancels, 2-Membership Renews
PROD_REBILL_METRIC	M - Month D - Day Y- Year
PROD_REBILL_PERIOD	7 30 90 etc
TAX_AMT	Used to capture the tax amount charged on a transaction. - Must be accompanied by TAX_EXEMPT=”N” for the amount to be calculated into the total transaction amount
TAX_EXEMPT	A flag to indicate if a business is exempted from paying taxes or not. (Y/N)
TRANS_TRIAL_REBILL_CUSTOMER_CONSENT	Reports on Customer Consent for rebill
TRANS_CUSTOMER_RECEIPT	Stores Customer transaction receipt
MBSHP_ID_XTL	For capturing customer membership id for memberships managed by the merchant.
TRANS_REBILL_TYPE	Reports if rebill is of either of these types NONE, TRIAL, INITIAL, REBILL
TPPE_ID	Third-Party Processing Entity: this can be an e-Commerce platform, a CRM, another gateway or any other entity which transmits cardholder data on a merchant’s behalf.
PMT_WALLET	This field shall be used to hold all the different types of wallets including e-wallet and m-wallet when available
PMT_WALLET_CRYPTOGRAM	Base64 URL encoded field that’ll hold the payload of the authorized transaction which comes from the Apple Pay or Google Pay session on the payment form hosted by the merchant
ORIG_CARD_BRAND_TRANSID	This is the incoming card scheme transaction id from the original transaction for the card and the merchant account.

4.7 Membership Adjustment Parameters

When Inovio is managing your memberships/subscriptions, below are the parameters needed for any adjustments and or cancellations.

Field Name	Description
REQUEST_REF_MBSHP_ID	Referring Membership ID

SUB_UPDATE_PROD_ID	New Subscription Product ID: Used for updating the current product ID of a membership record.
SUB_CANCEL_TYPE	Subscription Cancel Request Type
SUB_UPDATE_PMT_ID	Used to Update Card used for Membership (tied to specific Membership)
SUB_UPDATE	Request_Action for any membership update
SUB_UPDATE_TYPE	Immediately change a customer's membership/subscription from Product A to Product B and run a (Payment Type) AUTHCAP for Product B. - Optional

4.8 Ephemoral Tokenization

Rather than submitting the credit card number (PMT_ID) with a transaction, a token ID may be used instead (TOKEN_GUID). This unique ID is acquired by sending a request to our token service. The procedure for tokenization use is:

1) Submit a request to the token service (AJAX approach)
2) Receive the unique token ID
3) Submit a transaction request to the gateway, using TOKEN_GUID

! - *The token is unique, and for one-time use. A new token must be requested for each transaction.*

In order to utilize the enhanced tokenization parameters, please contact your gateway support representative to obtain your key.

4.8.1 Requesting the ephemoral token

The request for the unique token ID should be sent to the following URL:

https://api.inoviopay.com/payment/token_service.cfm?

4.8.1.1 Ephemoral Token Header Request Parameters

Field Name	Description	Additional Notes	Versioning
x-timestamp	The timestamp is open for five minutes (300 seconds) before the current time. Format: YYYYmmddHHMMSS	HH: Uses 24 hour clock HHMMSS: uses UTC timezone	4.9 and above
x-signature	Client's generated HMAC_SHA256 Base16 signature <x-timestamp><unique_id><site_id>	Generated using x-timestamp, unique_id, card_pan and site_id with the secret key assigned by Client Support.	4.9 and above

4.8.1.2 Ephemoral Token Body Request Parameters

Field Name	Description	Additional Notes	Versioning
card_pan	Customer card number to be tokenized		All
request_response_format	XML or JSON	Optional: Default to JSON	All

request_api_version	API Version of this document.		4.9 and above
site_id	Merchant's website ID		4.9 and above
unique_id	Alphanumeric ID linked to the request.	Generated by the client. Max of 32 characters.	4.9 and above

Ephemoral Token Request Example

https://api.inoviopay.com/payment/token_service.cfm?CARD_PAN=4111111111111111&REQUEST_RESPONSE_FORMAT=json

4.9+ version example: Secret key is Password123

X-SIGNATURE: 9cdc12c09f2fdb114167cad6fa451aeaa37c4df446720cae9805bf1e60e0603f

X-TIMESTAMP: 20250129225200

https://api.inoviopay.com/payment/token_service.cfm?CARD_PAN=4012345678901234&REQUEST_API_VERSION=4.14&UNIQUE_ID=32123456789012345678901234567890&SITE_ID=321&response_request_format=json

4.8.1.3 Ephemoral Token Response Parameters

4.8.1.3.1 Header Response Fields

Field Name	Description	Additional Notes	Versioning
x-timestamp	Format: YYYYMMDDHHMMSS	HH: Uses 24 hour clock HHMMSS: uses UTC timezone	4.9 and above
x-signature	Inovio's generated HMAC_SHA256 Base16 signature <x-timestamp><token_reqid><JSON body response>	Generated using x-timestamp, token reqid, and the full raw JSON body response including brackets with the secret key being assigned by Client Support.	4.9 and above

4.8.1.3.2 Body Response Fields

Field Name	Description	Additional Notes	Versioning
token_guid	Unique token ID which should subsequently be sent in the authorization request		All
token_ip	IP address of token request		All
token_reqid	Incremental ID of the token request		All

card_brand_name	Credit Card Network/Brand Name		4.9 and above
card_type	Indicates the card product sub category	i.e.: VISA Business	4.9 and above
card_bank	Issuer Bank name for the BIN		4.9 and above
card_country	Issuer Bank country for the BIN	Three character alpha country code	4.9 and above
card_account_fund_source	Identifies the source of the funds associated with the primary account for the card		4.9 and above
card_class	Categorizes the BIN as a Business, Corporate, Purchase, or Consumer card		4.9 and above

Ephemoral Token Response Example

```
{
"TOKEN_GUID": "7BA39EAFDAAD6B3FA8A974098A267258E6D622D9",
"TOKEN_IP": "10.13.100.134",
"TOKEN_REQID": "4283012"
}
```

4.9+ version example:

X-SIGNATURE: 35dab7703e959d363da14674a4f042b19b025c696da7af1325fb630bd342bae1

X-TIMESTAMP: 20241209230307

```
{
"TOKEN_GUID": "7BA39EAFDAAD6B3FA8A974098A267258E6D622D9",
"CARD_BRAND_NAME": "VISA",
"CARD_TYPE": "VISA TRADITIONAL",
"CARD_BANK": "BANK OF Inovio",
"CARD_COUNTRY": "USA",
"CARD_ACCOUNT_FUND_SOURCE": "Credit",
"CARD_CLASS": "CONSUMER",
"TOKEN_IP": "10.13.100.134",
"TOKEN_REQID": "4283012"
}
```

4.8.2 CCAUTHCAP Request Using an Ephemoral Tokenization

Example:

```
https://api.inoviopay.com/payment/pmt_service.cfm?request_action=CCAUTHCAP&request_api_version=4.14&req_username=test@example.com&req_password=Example0905&site_id=1111&request_response_format=JSON&li_value_1=10&li_prod_id_1=1001&TOKEN_GUID=7BA39EAFDAAD6B3FA8A974098A267258E6D622D9&PMT_KEY=123&PMT_EXPIRY=082025&MERCH_ACCT_ID=100&CUST_FNAME=Mister&CUST_LNAME=Customer&BILL_ADDR=123MainSt.&BILL_ADDR_CITY=LosAngeles&BILL_ADDR_STATE=CA&BILL_ADDR_ZIP=90032&BILL_ADDR_COUNTRY=US&CUST_EMAIL=test@test.com&REQUEST_CURRENCY=USD
```

5 Credit Card Transactions

5.1 Authorization

Authorization request is used to confirm the availability of funds in the cardholder's bank account. This type of transaction places a temporary hold or pending "auth" in the cardholder's bank account and does not guarantee payment. For this type of transaction, merchants must send service request action, "CCAUTHORIZE", in the transaction request.

Authorization request is the first part of a two-stage process of "authorizing" and "capturing" funds. This two stage process is commonly used by merchants who have to do partial fulfillments of orders. In order to capture the *Authorization* request, use Service Request Action, "CCCAPTURE" (see Section 5.2 for instructions).

Please allow a wait time of 120 seconds for CCAUTHORIZE, CCAUTHCAP and other requests. Even though the gateway usually adds only less than a second, downstream processors can take much longer to get a response from issuing banks.
***Note** that if the merchant chooses to time out their request earlier than 120 Seconds, then:*
a. The gateway is not responsible for lost transactions, transaction records and other failed functions
b. The merchant may choose to use CCSTATUS at a later time to find out the final outcome of their request (XTL_ORDER_ID would have to have been provided in the original request)
c. Webhooks/postbacks may or may not trigger at the time that the request is finalized at the gateway, and should not be relied upon.
Contact gateway support if you have questions.

5.1.1 Authorization Parameters

The table below only includes typical parameters sent in a simple Authorization transaction request. Note that merchants can send additional parameters listed in Section 4 of this specification.

Field Name	Description	Data Type	Requirement
REQUEST_ACTION	Service Request Action (Send "CCAUTHORIZE").	Service Request Types	Required
REQ_USERNAME	Service Request Username	Alphanumeric and Special Characters	Required
REQ_PASSWORD	Service Request Password	Alphanumeric and Special Characters	Required
REQUEST_RESPONSE_FORMAT	Service Response Format	Accepted values: "XML", "PIPES" and "JSON"	Optional (default: XML)
REQUEST_API_VERSION	API Version	Numeric	Required

SITE_ID	Merchant's Website ID	Numeric	Required
CUST_FNAME	Cardholder's First Name	Alphanumeric and Special Characters	Optional
CUST_LNAME	Cardholder's Last Name	Alphanumeric and Special Characters	Optional
CUST_EMAIL	Cardholder's Email Address	Alphanumeric and Special Characters	Optional
LI_COUNT_1	Line Item Count Max value is "99".	Numeric	Required
LI_PROD_ID_1	Line Item Product ID 1	Numeric	Required
LI_VALUE_1	Line Item Transaction Amount 1	Numeric	Required
XTL_ORDER_ID	Merchant's Order ID	Alphanumeric	Optional
BILL_ADDR	Cardholder Billing Street Address	Alphanumeric	Optional (address fields may be required by the bank)
BILL_ADDR_CITY	Cardholder's Billing City	Alphanumeric	Optional (address fields may be required by the bank)
BILL_ADDR_STATE	Cardholder's Billing State	2-letter State or Territory Code	Optional (some processors may require a valid 2-letter state or territory code)
BILL_ADDR_ZIP	Cardholder's Billing Postal/ZIP code	Alphanumeric	Optional (address fields may be required by the bank)
BILL_ADDR_COUNTRY	Cardholder's Billing Country	2-letter Country Code ISO 3166-1 alpha-2	Optional (address fields may be required by the bank)
PMT_NUMB	Credit Card Number	Numeric	Required
TOKEN_GUID	Ephemoral Token ID	Alphanumeric	Token ID used in place of pmt_numb. See Ephemoral Tokenization.

PMT_KEY	Credit Card CVV2 or CVC2 Code	Numeric (4)	Required
PMT_EXPIRY	Credit Card Expiration Date	Numeric MMYYYY Example: “122014”	Required
CUST_LOGIN	Cardholder’s Login or User Name	Alphanumeric and Special Characters	Optional
CUST_PASSWORD	Cardholder’s Password Password must be at least 10 characters with 1 number, lower case and upper case letter.	Alphanumeric	Optional
MERCH_ACCT_ID	Merchant Account ID	Numeric	If null, the system will follow merchant’s bank settings.
REQUEST_CURRENCY	3-letter Currency Code	Example: USD	Required
PMT_DESCRIPTOR	Dynamic Descriptor	Alphanumeric	Optional
PMT_DESCRIPTOR_PHONE	Bank Dynamic Customer Support Phone Number	Numeric	Optional
PMT_DESCRIPTOR_CITY	Bank Dynamic Customer Support City Applicable to MasterCard only	Alpha	Optional
CUST_PHONE	Cardholder’s Phone Number	Numeric	Optional
REQUEST_AFF_ID	External Affiliate ID	Alphanumeric	Optional
REQUEST_AFF_ID_SUB	External Sub-affiliate ID	Alphanumeric	Optional
UNIQUE_XTL_ORDER_ID	Enforces External Order ID uniqueness.	Possible values: “0” – disables flag “1” – decline request “2” – returns **Approval** response	Optional

PMT_ID_XTL	External Payment Unique Identifier	Alphanumeric (max length: 64)	Optional
MBSHP_ID_XTL	External Membership D	Alphanumeric	Optional (this will be require_ if the TRANS_REBILL_TYPE is not NONE)
TRANS_REBILL_TYPE	Rebill Type	One of NONE, TRIAL, INITIAL, REBILL	Optional
REQUEST_INITATOR	CIT (C) and MIT (M)	Used to identify transaction origination	Optional
REQUEST_INSTALLMENT	O (not used), 1 (used)	Used to denote if the rebill is an installment with start and end date.	Optional
PMT_NUMB_COF	O (not used), 1 (used)	Used to denote use of payment number stored by Merchant	Optional
REQUEST_REBILL	1 (yes for rebill), 2(Start Subscription), not used	Used to denote if the transaction is subscription card storage, a rebill or not used at all	Optional
TRANS_TRIAL_REBILL_CUSTOMER_CONSENT	Customer Consent for rebill after trial period	encoded character content	Required for the _first_ transaction of type REBILL following a transaction of type TRIAL on the same MBSHP_ID_XTL. Applies to MCC 5968 and Card_Brand_Name= Mastercard.
TRANS_CUSTOMER_RECEIPT	Payment Receipt	should accept encoded character content either freeform or html or xml content representing the actual receipt document sent to the customer	Optional but subsequent transactions on same MBSHP_ID_XTL unless previous trans are receipted. Applies to Card_Brand_Name = VISA. Also applies to MCC 5968 and Card_Brand_Name = Mastercard
CARD_ON_FILE_FLAG	Flags if COF or New PMT Card	possible values are 0 (COF) or 1(New)	Optional

5.1.2 Authorization Request Example

```
POST /payment/pmt_service.cfm HTTP/1.0
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1)
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash,
application/vnd.ms-powerpoint, application/vnd.ms-excel, application/msword, */*
Accept-Encoding: identity
Accept-Language: en-us,en
Host: Host.example.com
Accept-Charset: iso-8859-1,*,utf-8
Content-Type: application/x-www-form-urlencoded
Content-Length: 531
req_password=Test1234567&site_id=0&cust_fname=Daenarys&bill_addr_city=Los%20Angeles&b
ill_addr_state=CA&pmt_expiry=10%2F2020&xtl_cust_id=testcust11&pmt_key=123&request_res
ponse_format=XML&pmt_numb=4111111111111111&cust_lname=Targaryen&request_api_version=4
.14&cust_email=useremail%40tests.com&li_value_1=1.25&req_username=merchant100%40examp
le.com&li_prod_id_1=1001&request_currency=USD&xtl_order_id=testorder117&bill_addr_zip
=90401&request_action=CCAUTHORIZE&merch_acct_id=100&bill_addr=1 Main St
&bill_addr_country=US&pmt_id_xtl=ABCD12345&mbship_id_xtl=888920&card_on-
file_flag=0&trans_rebill_type=rebill
```

5.1.3 Authorization Response Example

```
<RESPONSE>
  <REQUEST_ACTION>CCAUTHCAP</REQUEST_ACTION>
  <TRANS_STATUS_NAME>APPROVED</TRANS_STATUS_NAME>
  <TRANS_VALUE>1.25</TRANS_VALUE>
  <TRANS_VALUE_SETTLED>1.25</TRANS_VALUE_SETTLED>
  <CURR_CODE_ALPHA_SETTLED>USD</CURR_CODE_ALPHA_SETTLED>
  <TRANS_EXCH_RATE/>
  <TRANS_ID>989898</TRANS_ID>
  <CUST_ID>212121</CUST_ID>
  <XTL_CUST_ID/>
 <REQ_ID>30814993221</REQ_ID>
  <PO_ID>17171717</PO_ID>
  <XTL_ORDER_ID/>
  <BATCH_ID>3114</BATCH_ID>
  <PROC_NAME>Test Processor</PROC_NAME>
  <MERCH_ACCT_ID>100</MERCH_ACCT_ID>
  <CARD_BRAND_NAME>Visa</CARD_BRAND_NAME>
  <CARD_DETAIL>CREDIT</CARD_DETAIL>
  <CARD_TYPE>VISA BUSINESS CARD</CARD_TYPE>
 <PMT_AAU_UPDATE_DT> 2019-07-17</PMT_AAU_UPDATE_DT>
 < PMT_AAU_UPDATE_DESC> Account Number Change Message</PMT_AAU_UPDATE_DESC>
  <CARD_CLASS>CONSUMER CREDIT</CARD_CLASS>
  <CARD_PREPAID>1</CARD_PREPAID>
  <CARD_BANK>BANK OF EUROPE</CARD_BANK>
  <CARD_COUNTRY>CRI</CARD_COUNTRY>
  <CARD_DETAIL>Credit</CARD_DETAIL>
  <CARD_BALANCE/>
  <PMT_L4>1111</PMT_L4>
  <PMT_ID>101011</PMT_ID>
  <PMT_ID_XTL/>
  <PROC_UDF01/>
  <PROC_UDF02/>
```

```
  <PROC_AUTH_RESPONSE>72682</PROC_AUTH_RESPONSE>
  <PROC_RETRIEVAL_NUM>74C8C7C8-5652-48BB-A68BGRACE46A2D1</PROC_RETRIEVAL_NUM>
  <PROC_REFERENCE_NUM/>
  <PROC_REDIRECT_URL/>
  <AVS_RESPONSE>M</AVS_RESPONSE>
  <CVV_RESPONSE>M</CVV_RESPONSE>
  <REQUEST_API_VERSION>4.14</REQUEST_API_VERSION>
  <PO_LI_ID_1>8509920</PO_LI_ID_1>
  <PO_LI_COUNT_1>1</PO_LI_COUNT_1>
  <PO_LI_AMOUNT_1>1.25</PO_LI_AMOUNT_1>
  <PO_LI_PROD_ID_1>1001</PO_LI_PROD_ID_1>
  <MBSHP_ID_1/>
</RESPONSE>
```

5.1.4 Failed Authorization Attempt Response Example

```
<RESPONSE>
<REQUEST_ACTION>CCAUTHORIZE</REQUEST_ACTION>
<TRANS_STATUS_NAME>DECLINED</TRANS_STATUS_NAME>
<TRANS_VALUE>5.01</TRANS_VALUE>
<TRANS_ID>20005</TRANS_ID>
<CUST_ID>2</CUST_ID>
<XTL_CUST_ID>c777777777</XTL_CUST_ID>
<MERCH_ACCT_ID>100</MERCH_ACCT_ID>
<CARD_BRAND_NAME>Mastercard</CARD_BRAND_NAME>
<PMT_L4>5100</PMT_L4>
<PMT_ID/> 100002</PMT_ID>
<PMT_ID_XTL/>
<API_RESPONSE>0</API_RESPONSE>
<API_ADVICE> </API_ADVICE>
<SERVICE_RESPONSE>640</SERVICE_RESPONSE>
<SERVICE_ADVICE>Retry</SERVICE_ADVICE>
<PROCESSOR_RESPONSE>501</PROCESSOR_RESPONSE>
<PROCESSOR_ADVICE>Retry</PROCESSOR_ADVICE>
<INDUSTRY_RESPONSE>0</INDUSTRY_RESPONSE>
<INDUSTRY_ADVICE> </INDUSTRY_ADVICE>
<REF_FIELD/>
<REQ_ID>30814993200</REQ_ID>
<PROC_NAME>Test Processor</PROC_NAME>
<AVS_RESPONSE/>
<CVV_RESPONSE/>
<REQUEST_API_VERSION>4.14</REQUEST_API_VERSION>
</RESPONSE>
```

5.1.5 Service Declined Response Example

```
<RESPONSE>
<REQUEST_ACTION>CCAUTHORIZE</REQUEST_ACTION>
<TRANS_STATUS_NAME/>
<CUST_ID/>
<XTL_CUST_ID>c777777777</XTL_CUST_ID>
<MERCH_ACCT_ID>100</MERCH_ACCT_ID>
<CARD_BRAND_NAME/>
<PMT_L4/>
<PMT_ID/> 440002</PMT_ID>
```

```
<PMT_ID_XTL/>
<API_RESPONSE>113</API_RESPONSE>
<API_ADVICE>Invalid Data</API_ADVICE>
<SERVICE_RESPONSE>0</SERVICE_RESPONSE>
<SERVICE_ADVICE>Declined</SERVICE_ADVICE>
<PROCESSOR_RESPONSE>0</PROCESSOR_RESPONSE>
<PROCESSOR_ADVICE> </PROCESSOR_ADVICE>
<INDUSTRY_RESPONSE>0</INDUSTRY_RESPONSE>
<INDUSTRY_ADVICE> </INDUSTRY_ADVICE>
<REF_FIELD>cust_password</REF_FIELD>
<REQ_ID>30814993201</REQ_ID>
<PROC_NAME/>
<AVS_RESPONSE/>
<CVV_RESPONSE/>
<REQUEST_API_VERSION>4.14</REQUEST_API_VERSION>
</RESPONSE>
```

The example above was declined by the API due to an invalid data in the CUST_PASSWORD field (see API**_ADVICE** and **REF_FIELD** parameters). Service-declined requests will also have the "Declined" message in the **SERVICE_ADVICE** field.

5.2 Delayed Capture

As described in *Section 5.1: Authorization*, an *Authorization* does not guarantee payment. In order to capture the pending authorization, merchants must send a capture request of the *Authorization* to the Payment Service.

To send a *Delayed Capture* request, merchants must send "CCCAPTURE" in the request_action parameter. The Payment Service allows merchants to send multiple partial amount capture requests or "partial captures" against a single *Authorization*, as long as the original amount of the *Authorization* is not exceeded.

5.2.1 Delayed Capture Request

Send a capture request by keying off the Order ID (PO_ID) of the original authorization. Send the Order ID in the **REQUEST_REF_PO_ID** parameter.

CCCAPTURE Required Fields

SITE_ID
REQ_USERNAME
REQ_PASSWORD
REQUEST_ACTION = CCCAPTURE
REQUEST_RESPONSE_FORMAT
REQUEST_REF_PO_ID
LI_VALUE_1
REQUEST_API_VERSION

Important Note

Merchants may send a different amount in the LI_VALUE_1 parameter. However, the amount may not exceed the original authorization's total amount.

5.3 Authorization and Capture (Sale)

The difference between an *Authorization* only and *Authorization and Capture* request is that the latter will also request capture of the authorized funds in a single request to the Payment Service.

Use service request action, "CCAUTHCAP" for *Authorization and Capture* requests. The fields for CCAUTHCAP requests are the same as CCAUTHORIZE requests described in Section 5.1.1: Authorization Parameters.

5.3.1 Authorization Request Example (CCAUTHCAP)

```
POST /payment/pmt_service.cfm HTTP/1.0
User-Agent: Mozilla/4.0 (compatible; MSIE 11.0; Windows NT 5.1)
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash,
application/vnd.ms-powerpoint, application/vnd.ms-excel, application/msword, */*
Accept-Encoding: identity
Accept-Language: en-us,en
Host: Host.example.com
Accept-Charset: iso-8859-1,*,utf-8
Content-Type: application/x-www-form-urlencoded
Content-Length: 53
req_username=test@example.com&req_password=Passw0rd!1&request_action=CCAUTHCAP&si
te_id=0&merch_acct_id=100&cust_fname=John&cust_lname=Doe&cust_email=user5@example.c
om&cust_login=username1&cust_password=12345678Xx&xtl_cust_id=c777777777&xtl_order_id=
o111111111&li_prod_id_1=1001&li_value_1=19.95&li_count_1=1&bill_addr=123 Main Street Apt.
1&bill_addr_city=Los
Angeles&bill_addr_state=CA&bill_addr_zip=90066&bill_addr_country=US&pmt_numb=51051051
05105100&pmt_key=123&pmt_expiry=12/2025&request_response_format=XML&xtl_ip=10.00.00
0.90&request_api_version=4.14
```

5.4 Reversal

5.4.1 Reversing or voiding an Authorization (CCREVERSE).

Merchants may request to reverse the original authorization by sending CCREVERSE in the request_action parameter.

To do a Reversal request, send the Order ID (PO_ID) of the original authorization in the **REQUEST_REF_PO_ID** parameter.

CCREVERSE Required Fields

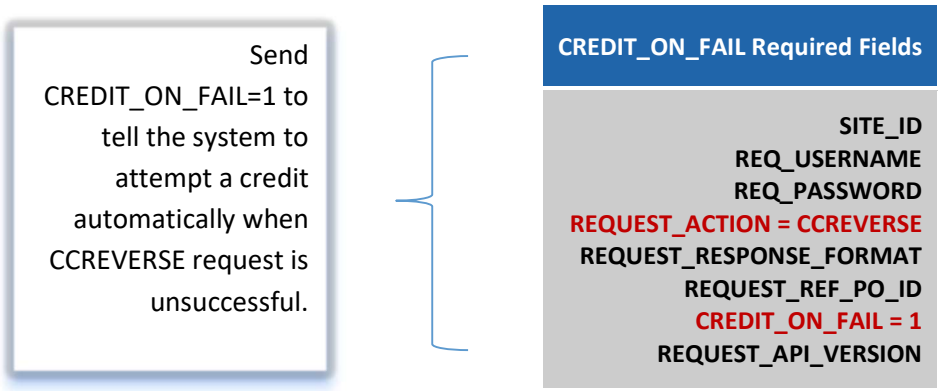
SITE_ID
REQ_USERNAME
REQ_PASSWORD
REQUEST_ACTION = CCREVERSE
REQUEST_RESPONSE_FORMAT
REQUEST_REF_PO_ID
REQUEST_API_VERSION

Important Note

Partial amount reversal requests are not supported by all processors. Please contact Client Services before implementing partial amount requests.

5.4.2 CREDIT_ON_FAIL Flag

In any case when the reversal request has failed, the system can automatically attempt a credit without the merchant sending a separate credit request. To enable this feature, merchants must set the CREDIT_ON_FAIL parameter to "1" and send this parameter with the CCREVERSE request. Set this field to null to disable.



The merchant should expect a *CCREVERSE* response if the reversal request is successful; if not successful, the API will return a CCCREDIT response instead.

5.4.2.1 Example Successful CCREVERSE Response

```
<RESPONSE>
<REQUEST_ACTION>CCREVERSE</REQUEST_ACTION>
<TRANS_STATUS_NAME>APPROVED</TRANS_STATUS_NAME>
<TRANS_VALUE>-1.22</TRANS_VALUE>
<CURR_CODE_ALPHA>USD</CURR_CODE_ALPHA>
<TRANS_VALUE_SETTLED>-1.22</TRANS_VALUE_SETTLED>
<CURR_CODE_ALPHA_SETTLED>USD</CURR_CODE_ALPHA_SETTLED>
<TRANS_EXCH_RATE/>
<TRANS_ID>69462</TRANS_ID>
<CUST_ID>1547</CUST_ID>
<XTL_CUST_ID/>
<PO_ID>261148</PO_ID>
<XTL_ORDER_ID/>
<BATCH_ID>1102</BATCH_ID>
<PROC_NAME>Test Processor</PROC_NAME>
<MERCH_ACCT_ID>100</MERCH_ACCT_ID>
<CARD_BRAND_NAME>Visa</CARD_BRAND_NAME>
<CARD_DETAIL>CREDIT</CARD_DETAIL>
<CARD_TYPE>VISA BUSINESS</CARD_TYPE>
<CARD_CLASS>CONSUMER CREDIT</CARD_CLASS>
<CARD_COUNTRY>CRI</CARD_COUNTRY>
<CARD_PREPAID>0</CARD_PREPAID>
<CARD_BANK/>
<CARD_BALANCE/>
<PMT_L4>1111</PMT_L4>
<PMT_ID/>
<PMT_ID_XTL/>
<PROC_UDF01/>
<PROC_UDF02/>
```

```
<PROC_AUTH_RESPONSE>74282</PROC_AUTH_RESPONSE>
<PROC_RETRIEVAL_NUM>TESTRETRIEVE123</PROC_RETRIEVAL_NUM>
<PROC_REFERENCE_NUM/>
<AVS_RESPONSE>M</AVS_RESPONSE>
<CVV_RESPONSE>M</CVV_RESPONSE>
<REQUEST_API_VERSION>4.14</REQUEST_API_VERSION>
<PO_LI_ID_1>300383</PO_LI_ID_1>
<PO_LI_COUNT_1>1</PO_LI_COUNT_1>
<PO_LI_AMOUNT_1>-1.22</PO_LI_AMOUNT_1>
<PO_LI_PROD_ID_1>1001</PO_LI_PROD_ID_1>
</RESPONSE>
```

5.4.2.2 Example Successful CCCREDIT Response

```
<RESPONSE>
<REQUEST_ACTION>CCCREDIT</REQUEST_ACTION>
<TRANS_STATUS_NAME>APPROVED</TRANS_STATUS_NAME>
<TRANS_VALUE>-1.22</TRANS_VALUE>
<CURR_CODE_ALPHA>USD</CURR_CODE_ALPHA>
<TRANS_VALUE_SETTLED>-1.22</TRANS_VALUE_SETTLED>
<CURR_CODE_ALPHA_SETTLED>USD</CURR_CODE_ALPHA_SETTLED>
<TRANS_EXCH_RATE/>
<TRANS_ID>21089</TRANS_ID>
<CUST_ID>1503</CUST_ID>
<XTL_CUST_ID/>
<PO_ID>215257</PO_ID>
<XTL_ORDER_ID/>
<BATCH_ID>1000</BATCH_ID>
<PROC_NAME>Test Processor</PROC_NAME>
<MERCH_ACCT_ID>100</MERCH_ACCT_ID>
<CARD_BRAND_NAME>Visa</CARD_BRAND_NAME>
<CARD_DETAIL>CREDIT</CARD_DETAIL>
<CARD_TYPE>VISA CLASSIC</CARD_TYPE>
<CARD_CLASS>CONSUMER CREDIT</CARD_CLASS>
<CARD_COUNTRY>CRI</CARD_COUNTRY>
<CARD_PREPAID>1</CARD_PREPAID>
<CARD_BANK/>
<CARD_BALANCE/>
<PMT_L4>2244</PMT_L4>
<PMT_ID/>
<PMT_ID_XTL/>
<PROC_UDF01/>
<PROC_UDF02/>
<PROC_AUTH_RESPONSE>33086</PROC_AUTH_RESPONSE>
<PROC_RETRIEVAL_NUM>B515URX2-6GRACE123</PROC_RETRIEVAL_NUM>
<PROC_REFERENCE_NUM/>
<AVS_RESPONSE>M</AVS_RESPONSE>
<CVV_RESPONSE>M</CVV_RESPONSE>
<REQUEST_API_VERSION>4.14</REQUEST_API_VERSION>
<PO_LI_ID_1>251038</PO_LI_ID_1>
<PO_LI_COUNT_1>1</PO_LI_COUNT_1>
<PO_LI_AMOUNT_1>-1.22</PO_LI_AMOUNT_1>
<PO_LI_PROD_ID_1>1001</PO_LI_PROD_ID_1>
<MBSHP_ID_1/>
```

```
</RESPONSE>
```

5.5 Credit

5.5.1 Minimum required parameters for issuing a Credit (CCCREDIT):

Merchants may request to return a captured authorization by sending CCCREDIT in the request_action parameter.

To do a Credit request, send the Order ID (PO_ID) of the original authorization in the **REQUEST_REF_PO_ID** parameter.

CCCREDIT Required Fields
SITE_ID
REQ_USERNAME
REQ_PASSWORD
REQUEST_ACTION = CCCREDIT
REQUEST_RESPONSE_FORMAT
REQUEST_REF_PO_ID
LI_VALUE_1
REQUEST_API_VERSION

Important Note

Merchants may only credit transactions that have been captured or settled.

A different amount from the original authorization maybe sent in the LI_VALUE_1 parameter. However, the amount may not exceed the original authorization's total amount. The exception is a force credit.

5.5.2 Force Credit Request

Force Credit is a type of transaction where the credit request is sent directly to the settlement file. Merchants do not need to send the REQUEST_REF_PO_ID parameter on this type of request. However, the FORCE_CREDIT=1 parameter is required, together with the full credit card information.

Important Note

Force Credits are only available to certain Merchant Accounts. Please contact your gateway support representative for more information.

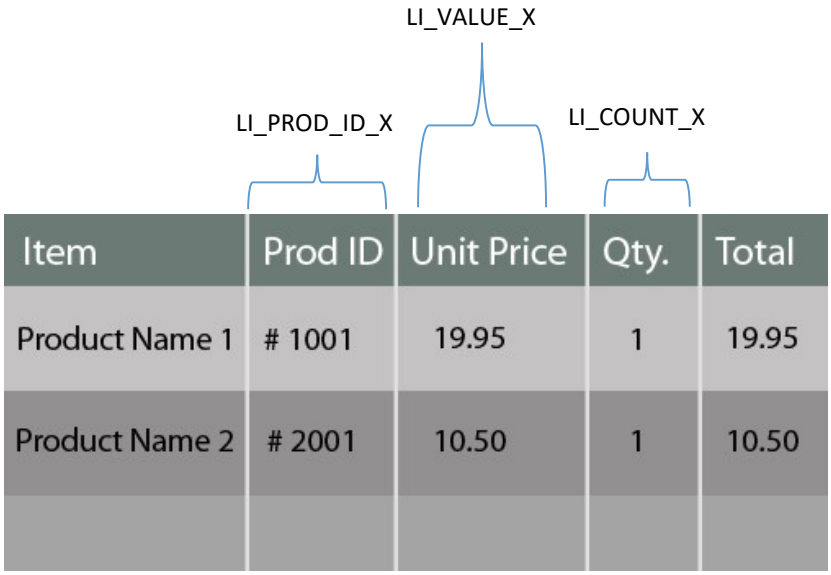
5.5.3 Multiple Line Items

Note: If your business does not intend to support purchase orders with more than one line item. Please skip this section. For more information, contact your gateway support representative.

The Payment Service supports purchase orders with more than one line item information. This feature will be useful for merchants using shopping cart like business models.

Merchants may use this feature by sending the Line Item Parameters: LI_PROD_ID_X, LI_COUNT_X, and LI_VALUE_X."X" indicates that this value is dynamic and should be replaced by a number depending on how many line items the merchant wants to send (i.e. for 2 line items the request should contain: LI_VALUE_1, LI_PROD_ID_1, LI_COUNT_1, LI_VALUE_2, LI_PROD_ID_2, and LI_COUNT_2).

5.5.4 Example CCAUTHORIZE request with Multiple Line Items.



The request for the shopping items above should look like:

```
/pmt_service.cfm?req_username=test@example.com&req_password=TestPassword1&request_action=CCAUTHORIZE&request_api_version=4.14&site_id=0&merch_acct_id=100&cust_fname=John&cust_lname=Doe&cust_email=test@example.com&cust_login=username1&cust_password=12345678Xx&bill_addr=1 Main Street&bill_addr_city=Hollywood&bill_addr_state=CA&bill_addr_zip=90078&bill_addr_country=US&pmt_numb=5105105105105100&pmt_key=123&pmt_expiry=12/2025&request_response_format=XML&xtl_ip=10.00.000.99&li_prod_id_1=1001&li_value_1=19.95&li_count_1=1&li_prod_id_2=2001&li_value_2=10.50&li_count_2=1
```

Note

Replace "CCAUTHORIZE" with "CCAUTHCAP" in the request above to do a CCAUTHCAP request instead of CCAUTHORIZE.

5.5.5 Example CCAUTHORIZE Response with Multiple Line Items

The Line Item response parameters below are highlighted in gray.

```
<RESPONSE>
<REQUEST_ACTION>CCAUTHORIZE</REQUEST_ACTION>
<TRANS_STATUS_NAME>APPROVED</TRANS_STATUS_NAME>
<CURR_CODE_ALPHA>USD</CURR_CODE_ALPHA>
<TRANS_VALUE>30.45</TRANS_VALUE>
<CURR_CODE_ALPHA>USD</CURR_CODE_ALPHA>
<TRANS_VALUE_SETTLED>30.45</TRANS_VALUE_SETTLED>
```

```
<CURR_CODE_ALPHA_SETTLED>USD</CURR_CODE_ALPHA_SETTLED>
<TRANS_EXCH_RATE/>
<TRANS_ID>30000</TRANS_ID>
<CUST_ID>2000</CUST_ID>
<XTL_CUST_ID/>
<PO_ID>77777</PO_ID>
<XTL_ORDER_ID/>
<BATCH_ID>1001</BATCH_ID>
<PROC_NAME>Test Processor</PROC_NAME>
<MERCH_ACCT_ID>100</MERCH_ACCT_ID>
<CARD_BRAND_NAME>Visa</CARD_BRAND_NAME>
<CARD_DETAIL>CREDIT</CARD_DETAIL>
<CARD_TYPE>VISA BUSINESS</CARD_TYPE>
<CARD_CLASS>CONSUMER CREDIT</CARD_CLASS>
<CARD_COUNTRY>CRI</CARD_COUNTRY>
<CARD_PREPAID>0</CARD_PREPAID>
<CARD_BANK/>
<CARD_BALANCE/>
<PMT_L4>1111</PMT_L4>
<PMT_ID/>
<PMT_ID_XTL/>
<PROC_UDF01/><PROC_UDF02/>
<PROC_AUTH_RESPONSE>51402</PROC_AUTH_RESPONSE>
<PROC_RETRIEVAL_NUM>7169CEF3-64EF-46BB-55A333956994D2D</PROC_RETRIEVAL_NUM>
<PROC_REFERENCE_NUM/>
<REQ_ID>30814003200</REQ_ID>
<AVS_RESPONSE>M</AVS_RESPONSE>
<CVV_RESPONSE>M</CVV_RESPONSE>
<REQUEST_API_VERSION>4.14</REQUEST_API_VERSION>
<PO_LI_ID_1>123455</PO_LI_ID_1>
<PO_LI_COUNT_1>1</PO_LI_COUNT_1>
<PO_LI_AMOUNT_1>19.95</PO_LI_AMOUNT_1>
<PO_LI_PROD_ID_1>1001</PO_LI_PROD_ID_1>
<PO_LI_ID_2>123456</PO_LI_ID_2>
<PO_LI_COUNT_2>1</PO_LI_COUNT_2>
<PO_LI_AMOUNT_2>10.5</PO_LI_AMOUNT_2>
<PO_LI_PROD_ID_2>2001</PO_LI_PROD_ID_2>
<MBSHP_ID_1/>
</RESPONSE>
```

LINE ITEM RESPONSE

5.5.6 Example Request for Capturing a Specific Line Item

To do a CCCAPTURE request of a line item, send the Line Item ID you want to capture in the REQUEST_PO_LI_ID parameter.

* **REQUEST_REF_PO_ID** and **LI_VALUE_1** are also required.

Multiline Items: CCCAPTURE Required Fields

SITE_ID
REQ_USERNAME
REQ_PASSWORD
REQUEST_ACTION = CCCAPTURE
REQUEST_RESPONSE_FORMAT
REQUEST_REF_PO_LI_ID
LI_VALUE_1
REQUEST_REF_PO_ID

Merchants should only use the request_ref_po_li_id when sending a delayed capture request against a specific line item.

5.5.7 Capturing, Reversing or Crediting the Entire Order with Multiple Line Items

To capture, reverse, or credit the entire purchase order, just send the correct REQUEST_ACTION and REQUEST_REF_PO_ID with other required parameters in the API request.

5.5.7.1 Multiline Items: CCCAPTURE Example Request

Note that this request will capture the second line item from the CCAUTHORIZE example above:

```
/pmt_service.cfm?req_password=TestPassword1&site_id=0&request_ref_po_li_id=123456&req
uest_ref_po_id=77777&chkavs=F&request_response_format=JSON&req_username=test%40exampl
e.com&li_value_1=10.50&request_action=CCCAPTURE&request_api_version=4.14
```

5.5.7.2 Multiline Items: CCCAPTURE Example Approved Response

```
<RESPONSE>
<REQUEST_ACTION>CCCAPTURE</REQUEST_ACTION>
<TRANS_STATUS_NAME>APPROVED</TRANS_STATUS_NAME>
<TRANS_VALUE>10.5</TRANS_VALUE>
<CURR_CODE_ALPHA>USD</CURR_CODE_ALPHA>
<TRANS_VALUE_SETTLED>10.5</TRANS_VALUE_SETTLED>
<CURR_CODE_ALPHA_SETTLED>USD</CURR_CODE_ALPHA_SETTLED>
<TRANS_EXCH_RATE/>
<TRANS_ID>124447</TRANS_ID>
<CUST_ID>2000</CUST_ID>
<XTL_CUST_ID/>
<PO_ID>77777</PO_ID>
<XTL_ORDER_ID/>
<BATCH_ID>1101</BATCH_ID>
<PROC_NAME>Test Processor</PROC_NAME>
<MERCH_ACCT_ID>100</MERCH_ACCT_ID>
<CARD_BRAND_NAME>Visa</CARD_BRAND_NAME>
<CARD_DETAIL>CREDIT</CARD_DETAIL>
<CARD_TYPE>VISA BUSINESS</CARD_TYPE>
<CARD_CLASS>CONSUMER CREDIT</CARD_CLASS>
<CARD_COUNTRY>CRI</CARD_COUNTRY>
<CARD_PREPAID>0</CARD_PREPAID>
<CARD_BANK/>
```

```
<CARD_BALANCE/>
<PMT_L4>1111</PMT_L4>
<PMT_ID/>
<PMT_ID_XTL/>
<PROC_UDF01/><PROC_UDF02/>
<PROC_AUTH_RESPONSE/>
<PROC_RETRIEVAL_NUM>9287005b-0740-4049-8526-9dcc6f06e610</PROC_RETRIEVAL_NUM>
<PROC_REFERENCE_NUM>5478723</PROC_REFERENCE_NUM>
<AVS_RESPONSE/>
<CVV_RESPONSE/>
<REQUEST_API_VERSION>4.14</REQUEST_API_VERSION>
<PO_LI_ID_1>123456</PO_LI_ID_1>
<PO_LI_COUNT_1>1</PO_LI_COUNT_1>
<PO_LI_AMOUNT_1>10.5</PO_LI_AMOUNT_1>
<PO_LI_PROD_ID_1>2001</PO_LI_PROD_ID_1>
<MBSHP_ID/>
</RESPONSE>
```

5.5.8 Example Request for Sending Reversal of a Specific Line Item

To do a CCREVERSE request of a line item, send the Line Item ID you want to reverse in REQUEST_PO_LI_ID parameter.

* **REQUEST_REF_PO_ID** is also required.

Multiline Items: CCREVERSE Required Fields

SITE_ID
REQ_USERNAME
REQ_PASSWORD
REQUEST_ACTION = CCREVERSE
REQUEST_RESPONSE_FORMAT
REQUEST_REF_PO_LI_ID
REQUEST_REF_PO_ID
REQUEST_API_VERSION

5.5.8.1 Multiline Items: CCREVERSE Example Request

```
/pmt_service.cfm?req_password=TestPassword1&site_id=0&request_ref_po_li_id=123456&req
uest_ref_po_id=77777&chkavs=F&request_response_format=XML&req_username=test%40example
.com&li_value_1=10.50&request_api_version=4.14&request_action=CCREVERSE
```

5.5.8.2 Multiline Items: CCREVERSE Example Response

```
<RESPONSE>
<REQUEST_ACTION>CCREVERSE</REQUEST_ACTION>
<TRANS_STATUS_NAME>APPROVED</TRANS_STATUS_NAME>
<TRANS_VALUE>-10.50</TRANS_VALUE>
<CURR_CODE_ALPHA>USD</CURR_CODE_ALPHA>
<TRANS_VALUE_SETTLED>-10.50</TRANS_VALUE_SETTLED>
<CURR_CODE_ALPHA_SETTLED>USD</CURR_CODE_ALPHA_SETTLED>
```

```
<TRANS_EXCH_RATE/>
<TRANS_ID>100002</TRANS_ID>
<CUST_ID>2000</CUST_ID>
 <XTL_CUST_ID/>
<PO_ID>77777</PO_ID>
<XTL_ORDER_ID/>
<BATCH_ID>1001</BATCH_ID>
<PROC_NAME>Test Processor</PROC_NAME>
<MERCH_ACCT_ID>100</MERCH_ACCT_ID>
<MERCH_ACCT_ID>100</MERCH_ACCT_ID>
<CARD_BRAND_NAME>Visa</CARD_BRAND_NAME>
<CARD_TYPE>VISA BUSINESS</CARD_TYPE>
<CARD_CLASS>CONSUMER CREDIT</CARD_CLASS>
<CARD_COUNTRY>CRI</CARD_COUNTRY>
<CARD_PREPAID>0</CARD_PREPAID>
<CARD_BANK/>
<CARD_BALANCE/>
<PMT_L4>5100</PMT_L4>
<PMT_ID/>
<PMT_ID_XTL/>
<PROC_UDF01/><PROC_UDF02/>
<PROC_AUTH_RESPONSE>69803</PROC_AUTH_RESPONSE>
<PROC_RETRIEVAL_NUM>31B22CB-A453-GRA3CE5</PROC_RETRIEVAL_NUM>
<PROC_REFERENCE_NUM/>
<AVS_RESPONSE>M</AVS_RESPONSE>
<CVV_RESPONSE>M</CVV_RESPONSE>
<REQUEST_API_VERSION>4.14</REQUEST_API_VERSION>
<PO_LI_ID_1>123456</PO_LI_ID_1>
<PO_LI_COUNT_1>1</PO_LI_COUNT_1>
<PO_LI_AMOUNT_1>-10.50</PO_LI_AMOUNT_1>
<PO_LI_PROD_ID_1>2001</PO_LI_PROD_ID_1>
<MBSHP_ID/>
</RESPONSE>
```

5.5.8.3 Crediting a specific line item

To do a CCCREDIT request of a line item, send the Line Item ID you want to credit in REQUEST_PO_LI_ID parameter.

*** REQUEST_REF_PO_ID** and **LI_VALUE_1** are also required.

Multiline Items: CCCREDIT Required Fields

SITE_ID
REQ_USERNAME
REQ_PASSWORD
REQUEST_API_VERSION
REQUEST_ACTION = CCCREDIT
REQUEST_RESPONSE_FORMAT
REQUEST_REF_PO_LI_ID
LI_VALUE_1
REQUEST_REF_PO_ID

5.5.8.4 Multiline Items: CCCREDIT Example Request

```
/pmt_service.cfm?req_password=TestPassword1&site_id=0&request_ref_po_li_id=123456&req
uest_ref_po_id=77777&chkavs=F&request_response_format=XML&req_username=test%40example
.com&li_value_1=10.50&request_api_version=4.14&request_action=CCCREDIT
```

5.5.8.5 Multiline Items: CCCREDIT Example Response

```
<RESPONSE>
<REQUEST_ACTION>CCCREDIT</REQUEST_ACTION>
<TRANS_STATUS_NAME>APPROVED</TRANS_STATUS_NAME>
<TRANS_VALUE>-10.5</TRANS_VALUE>
<TRANS_ID>100002</TRANS_ID>
<CURR_CODE_ALPHA>USD</CURR_CODE_ALPHA>
<TRANS_VALUE_SETTLED>-10.5</TRANS_VALUE_SETTLED>
<CURR_CODE_ALPHA_SETTLED>USD</CURR_CODE_ALPHA_SETTLED>
<TRANS_EXCH_RATE/>
<CUST_ID>2000</CUST_ID>
<XTL_CUST_ID/>
<PO_ID>77777</PO_ID>
<XTL_ORDER_ID/>
<BATCH_ID>1001</BATCH_ID>
<PROC_NAME>Test Processor</PROC_NAME>
<MERCH_ACCT_ID>100</MERCH_ACCT_ID>
<CARD_BRAND_NAME>Visa</CARD_BRAND_NAME>
<CARD_TYPE>VISA BUSINESS</CARD_TYPE>
<CARD_CLASS>CONSUMER CREDIT</CARD_CLASS>
<CARD_COUNTRY>CRI</CARD_COUNTRY>
<CARD_DETAIL>Credit</CARD_DETAIL>
<CARD_PREPAID>0</CARD_PREPAID>
<CARD_BANK>BANK OF TEST BANKS</CARD_BANK>
<CARD_BALANCE/>
<PMT_L4>5100</PMT_L4>
<PMT_ID/>
<PMT_ID_XTL/>
<PROC_UDF01/><PROC_UDF02/>
<PROC_AUTH_RESPONSE>69803</PROC_AUTH_RESPONSE>
<PROC_RETRIEVAL_NUM>31B22CB-A453-GRA3CE5</PROC_RETRIEVAL_NUM>
<PROC_REFERENCE_NUM/>
<AVS_RESPONSE>M</AVS_RESPONSE>
<CVV_RESPONSE>M</CVV_RESPONSE>
<REQUEST_API_VERSION>4.14</REQUEST_API_VERSION>
<PO_LI_ID_1>123457</PO_LI_ID_1>
<PO_LI_COUNT_1>1</PO_LI_COUNT_1>
<PO_LI_AMOUNT_1>-10.5</PO_LI_AMOUNT_1>
<PO_LI_PROD_ID_1>2001</PO_LI_PROD_ID_1>
<MBSHP_ID/>
</RESPONSE>
```

A new Line Item ID is created for credit transactions.

5.5.9 Line Item Types

The Line Item Type is not returned in the gateway API response, however, this data will be available on some of the client reports. The table below lists different type of line items:

Line Item Type ID	Description

1	Goods and Services
2	Discount
3	Tax
4	Shipping
5	Chargeback
6	Refund

5.6 Checking the Status of an order

To check the status of an order, merchants may set the REQUEST_ACTION parameter to "CCSTATUS" along with the Order ID in the REQUEST_REF_PO_ID parameter.

If REQUEST_REF_PO_ID is not available, merchants may use the XTL_ORDER_ID that was sent in the original authorization instead. Send this value in the REQUEST_REF_PO_ID_XTL parameter.

5.6.1 CCSTATUS Example Request using REQUEST_REF_PO_ID

```
/pmt_service.cfm?req_password=P5ssword1&site_id=1234&request_ref_po_id=1234567&request
_response_format=JSON&req_username=test1%40example.net&request_api_version=4.14&reque
st_action=CCSTATUS
```

5.6.2 CCSTATUS Example Request using REQUEST_REF_PO_ID_XTL

```
/pmt_service.cfm?req_password=
P5ssword1&site_id=1234&request_ref_po_id_xtl=order123&request_response_format=JSON&req
_username=test1%40example.net&request_api_version=4.14&request_action=CCSTATUS
```

5.6.3 CCSTATUS Example Response

The response example below shows that the order was captured and then reversed. Merchants may refer to the TRANS_STATUS_NAME in checking the status of a specific transaction request.

```
<RESPONSE>
  <TRANSACTION TRANS_ID="43500">
    <REQUEST_ACTION>CCAUTHORIZE</REQUEST_ACTION>
    <TRANS_STATUS_NAME>APPROVED</TRANS_STATUS_NAME>
    <TRANS_VALUE>10</TRANS_VALUE>
    <CUST_ID>1500</CUST_ID>
    <XTL_CUST_ID/>
    <PO_ID>1234567</PO_ID>
    <XTL_ORDER_ID> order123</XTL_ORDER_ID>
    <BATCH_ID>1001</BATCH_ID>
    <PROC_NAME>Test Processor</PROC_NAME>
    <MERCH_ACCT_ID>100</MERCH_ACCT_ID>
    <CARD_BRAND_NAME>Visa</CARD_BRAND_NAME>
    <PMT_L4>3762</PMT_L4>
    <PROC_UDF01/>
```

```
    <PROC_UDF02/>
    <PROC_AUTH_RESPONSE>46177</PROC_AUTH_RESPONSE>
    <PROC_RETRIEVAL_NUM>56789AB</PROC_RETRIEVAL_NUM>
    <PROC_REFERENCE_NUM/>
    <AVS_RESPONSE>M</AVS_RESPONSE>
    <CVV_RESPONSE>M</CVV_RESPONSE>
  </TRANSACTION>
  <TRANSACTION TRANS_ID="43502">
    <REQUEST_ACTION>CCREVERSE</REQUEST_ACTION>
    <TRANS_STATUS_NAME>APPROVED</TRANS_STATUS_NAME>
    <TRANS_VALUE>-10</TRANS_VALUE>
    <CUST_ID>1500</CUST_ID>
    <XTL_CUST_ID/>
    <PO_ID>1234567</PO_ID>
    <XTL_ORDER_ID> order123</XTL_ORDER_ID>
    <BATCH_ID>1001</BATCH_ID>
    <PROC_NAME>Test Processor</PROC_NAME>
    <MERCH_ACCT_ID>100</MERCH_ACCT_ID>
<CARD_BRAND_NAME>Visa</CARD_BRAND_NAME>
<CARD_DETAIL>DEBIT</CARD_DETAIL>
<CARD_TYPE>VISA BUSINESS</CARD_TYPE>
<CARD_CLASS>CONSUMER CREDIT</CARD_CLASS>
<CARD_COUNTRY>CRI</CARD_COUNTRY>
<CARD_PREPAID>0</CARD_PREPAID>
<CARD_BANK/>
<REQ_ID>304993200</REQ_ID>
<PMT_L4>3762</PMT_L4>
<PMT_ID>43333</PMT_L4>
<PMT_ID_XTL/>
    <PROC_UDF01/>
    <PROC_UDF02/>
    <PROC_AUTH_RESPONSE>62088</PROC_AUTH_RESPONSE>
    <PROC_RETRIEVAL_NUM>123456789A</PROC_RETRIEVAL_NUM>
    <PROC_REFERENCE_NUM/>
    <AVS_RESPONSE>M</AVS_RESPONSE>
    <CVV_RESPONSE>M</CVV_RESPONSE>
  </TRANSACTION>
  <TRANSACTION TRANS_ID="43501">
    <REQUEST_ACTION>CCCAPTURE</REQUEST_ACTION>
    <TRANS_STATUS_NAME>APPROVED</TRANS_STATUS_NAME>
    <TRANS_VALUE>10</TRANS_VALUE>
    <CUST_ID>1500</CUST_ID>
    <XTL_CUST_ID/>
    <PO_ID>1234567</PO_ID>
    <XTL_ORDER_ID> order123</XTL_ORDER_ID>
    <BATCH_ID>1001</BATCH_ID>
    <PROC_NAME>Test Processor</PROC_NAME>
    <MERCH_ACCT_ID>100</MERCH_ACCT_ID>
<CARD_BRAND_NAME>Visa</CARD_BRAND_NAME>
<CARD_DETAIL>DEBIT</CARD_DETAIL>
<CARD_TYPE>VISA BUSINESS</CARD_TYPE>
<CARD_CLASS>CONSUMER CREDIT</CARD_CLASS>
<CARD_COUNTRY>CRI</CARD_COUNTRY>
<CARD_PREPAID>0</CARD_PREPAID>
```

```
<CARD_BANK/>
    <PMT_L4>3762</PMT_L4>
    <PROC_UDF01/>
    <PROC_UDF02/>
    <PROC_AUTH_RESPONSE>63106</PROC_AUTH_RESPONSE>
    <PROC_RETRIEVAL_NUM>F56789</PROC_RETRIEVAL_NUM>
    <PROC_REFERENCE_NUM/>
    <AVS_RESPONSE>M</AVS_RESPONSE>
    <CVV_RESPONSE>M</CVV_RESPONSE>
  </TRANSACTION>
</RESPONSE>
```

5.6.4 CCSTATUS Response Fields

- TRANS_ID – Unique transaction ID.
- REQUEST_ACTION – This will return the Service Request Action the merchant sent in the transaction request.
- TRANS_STATUS_NAME – Transaction Status.
- TRANS_VALUE – Total transaction amount for all line items.
- CUST_ID – Customer ID.
- XTL_CUST_ID- Merchant's Customer ID.
- PO_ID – Purchase order ID.
- XTL_ORDER_ID – Merchant's order ID.
- BATCH_ID – Settlement Batch ID.
- PROC_NAME – Merchant Processor Name (Example: "Payvision").
- MERCH_ACCT_ID - Merchant Bank's Account ID.
- CARD_BRAND_NAME – Bank account or credit card type (Example: "Mastercard").
- PMT_L4 – Payment account or credit card's last 4 digits.
- PMT_ID – Payment ID
- PMT_ID_XTL – External Payment ID
- PROC_UDF01 – Processor User Defined Field 1
- PROC_UDF02 – Processor User Defined Field 2
- PROC_AUTH_RESPONSE – Processor Authorization Response Code.
- PROC_RETRIEVAL_NUM – Processor Retrieval Number or GUID
- PROC_REFERENCE_NUM – Processor Reference Number.
- AVS_RESPONSE – Address Verification Service Response Code.
- CVV_RESPONSE – Card Verification Value Response Code.
- REQ_ID – Request Identifier
- REQUEST_API_VERSION – Payment Service API Version (4.14).
- PO_LI_ID_X – Purchase Order Line Item ID.
- PO_LI_COUNT_X – Purchase Order Line Item Count.
- PO_LI_AMOUNT_X – Purchase Order Line Item Total Amount.
- PO_LI_PROD_ID_X – Purchase Order Line Item Product ID.
- MBSHP_ID – Membership ID (returned on membership transactions)

5.6.5 Transaction Status is PENDING

Some authorization requests may get a transaction status of "Pending". This usually happens if the processor insists that the customer goes through an extra credit card authentication step such as 3-D Secure (i.e. Verified by Visa or Mastercard SecureCode). Merchants should also expect a

redirect URL in PROC_REDIRECT_URL field in the gateway response in case of 3-D Secure authentication.

ACH and European Direct Debit authorization requests will also return "Pending" status.

5.7 Updating Order Data

"CCTRANSUPDATE" can be used to update specific data elements of an order. Currently, only TRANS_CUST_RECEIPT is available to be updated.

To update an order with a receipt, merchants may set the REQUEST_ACTION parameter to "CCTRANSUPDATE" along a reference to the order, either in the REQUEST_REF_PO_ID (referencing the PO_ID from a previous order) or REQUEST_REF_PO_ID_XTL (referencing the XTL_ORDER_ID from a previous order) parameter.

5.7.1 CCTRANSUPDATE Example Request using REQUEST_REF_PO_ID

```
/pmt_service.cfm?req_password=P5ssword1&site_id=1234&request_ref_po_id=1234567&request_response
_format=JSON&req_username=test1%40example.net&request_api_version=4.14&request_action=
CCTRANSUPDATE&mbshp_id_xtl=1502&trans_customer_receipt=
aHR0cHM6Ly9udXRyaXN5cy5yZWMudHJhbnNfaWQyMzQ=
```

5.7.2 CCTRANSUPDATE Example Request using REQUEST_REF_PO_ID_XTL

```
https://api.inoviopay.com/payment/pmt_service.cfm ?req_password=
P5ssword1&site_id=1234&request_ref_po_id_xtl=order123&request_response_format=JSON&req_usernam
e=test1%40example.net&request_api_version=4.14&request_action=CCTRANSUPDATE&mbshp_id_xtl=150
2&trans_customer_receipt= aHR0cHM6Ly9udXRyaXN5cy5yZWMudHJhbnNfaWQyMzQ=
```

5.7.3 CCTRANSUPDATE Example Response

The response example below shows that the order was captured and then reversed. Merchants may refer to the TRANS_STATUS_NAME in checking the status of a specific transaction request.

```
<RESPONSE>
  <TRANSACTION TRANS_ID="43500">
    <REQUEST_ACTION>CCAUTHORIZE</REQUEST_ACTION>
    <TRANS_STATUS_NAME>APPROVED</TRANS_STATUS_NAME>
    <TRANS_VALUE>10</TRANS_VALUE>
    <CUST_ID>1500</CUST_ID>
    <XTL_CUST_ID/>
    <PO_ID>1234567</PO_ID>
    <XTL_ORDER_ID> order123</XTL_ORDER_ID>
    <BATCH_ID>1001</BATCH_ID>
    <PROC_NAME>Test Processor</PROC_NAME>
    <MERCH_ACCT_ID>100</MERCH_ACCT_ID>
    <CARD_BRAND_NAME>Visa</CARD_BRAND_NAME>
    <PMT_L4>3762</PMT_L4>
    <PROC_UDF01/>
    <PROC_UDF02/>
    <PROC_AUTH_RESPONSE>46177</PROC_AUTH_RESPONSE>
    <PROC_RETRIEVAL_NUM>56789AB</PROC_RETRIEVAL_NUM>
    <PROC_REFERENCE_NUM/>
    <AVS_RESPONSE>M</AVS_RESPONSE>
    <CVV_RESPONSE>M</CVV_RESPONSE>
```

```
  <MBSHP_ID_XTL>1502</MBSHP_ID_XTL>
</TRANSACTION>
```

5.7.4 CCTRANSUPDATE Response Fields

- TRANS_ID – Unique transaction ID.
- REQUEST_ACTION – This will return the Service Request Action the merchant sent in the transaction request.
- TRANS_STATUS_NAME – Transaction Status.
- TRANS_VALUE – Total transaction amount for all line items.
- CUST_ID – Customer ID.
- XTL_CUST_ID- Merchant's Customer ID.
- PO_ID – Purchase order ID.
- XTL_ORDER_ID – Merchant's order ID.
- BATCH_ID – Settlement Batch ID.
- PROC_NAME – Merchant Processor Name (Example: "Payvision").
- MERCH_ACCT_ID - Merchant Bank's Account ID.
- CARD_BRAND_NAME – Bank account or credit card type (Example: "Mastercard").
- PMT_L4 – Payment account or credit card's last 4 digits.
- PMT_ID – Payment ID
- PMT_ID_XTL – External Payment ID
- PROC_UDF01 – Processor User Defined Field 1
- PROC_UDF02 – Processor User Defined Field 2
- PROC_AUTH_RESPONSE – Processor Authorization Response Code.
- PROC_RETRIEVAL_NUM – Processor Retrieval Number or GUID
- PROC_REFERENCE_NUM – Processor Reference Number.
- AVS_RESPONSE – Address Verification Service Response Code.
- CVV_RESPONSE – Card Verification Value Response Code.
- REQ_ID – Request Identifier
- REQUEST_API_VERSION – Payment Service API Version (4.14).
- PO_LI_ID_X – Purchase Order Line Item ID.
- PO_LI_COUNT_X – Purchase Order Line Item Count.
- PO_LI_AMOUNT_X – Purchase Order Line Item Total Amount.
- PO_LI_PROD_ID_X – Purchase Order Line Item Product ID.
- MBSHP_ID – Membership ID (returned on membership transactions)
- MBSHP_ID_XTL – Customer membership ID received from the Merchant

6 ACH/eCheck Transactions

The following steps describe the lifecycle of a transaction processed with an ACH account:

1. Customer initiates purchase on the payment page.
2. Merchant submits transaction request to Payments Service. This should include the following data:
    a. Customer and payment information.
    b. Merchant gateway credentials and other gateway parameters as needed.
3. Payments Service transmits transaction request to the bank.
4. Merchant receives gateway response. Transaction status should be "Pending".
5. Post processing transaction status update (both methods are optional):

a. Postback API: Payments Service sends Postback to the merchant as the transaction ***state*** changes. Postback is real-time.
b. Order Detail Report: Merchant downloads order and transaction data using Order Detail Report service.

! If you choose not to use methods 5a or 5b to receive updates to the status of an ACH order, you can log into our portal and check the status by searching for the order ID.

6.1 Transaction State Updates Through Postback (Optional)

To keep merchants updated on the status of their pending ACH transactions, we can send real-time Postbacks to the merchant.

6.2 Transaction State using Order Detail API (Optional)

Merchants may also download transaction status updates as an alternative to Postback by using Order Detail Report service.

6.2.1 Transaction Status

TRANS_STATUS_NAME	Description
APPROVED	Transaction has been approved.
PENDING	Transaction is in pending status.
RUNNING	Transaction processing was not completed or is waiting completion usually because of gateway error.

6.3 Gateway Request Parameters

The table below describes parameters needed in sending ACH purchase request to the gateway. Merchants may send additional parameters, as described in the Payments Payment Service API, such as customer billing address, email, website login and other pass-through data.

Field Name	Description
REQUEST_ACTION	Service Request Action (Send “ACHAUTHCAP”)
REQ_USERNAME	Service Request Username
REQ_PASSWORD	Service Request Password
REQUEST_RESPONSE_FORMAT	Service Response Format (XML or JSON)
REQUEST_API_VERSION	API Version (example: “4.14”)
SITE_ID	Merchant’s Website ID
LI_PROD_ID_1	Line Item Product ID 1
LI_VALUE_1	Line Item Transaction Amount 1

PMT_NUMB	Bank Account Number
BANK_IDENTIFIER	Routing Number
MERCH_ACCT_ID	Merchant Account ID
REQUEST_CURRENCY	3-letter Currency Code
CUST_FNAME	Customer First Name
CUST_LNAME	Customer Last Name
BILL_ADDR	Billing Street Address
BILL_ADDR_CITY	Billing City Name
BILL_ADDR_STATE	Billing State 2-letter Code
BILL_ADDR_COUNTRY	Billing Country 2-letter Code
BILL_ADDR_ZIP	Billing ZIP or Postal Code

6.4 Gateway Actions

Below are the only supported gateway actions for ACH. Any other gateway action or feature will not work.

REQUEST_ACTION	Description
ACHAUTHCAP	Used for authorization and capture requests.
ACHAUTHORIZE	Used for Authorizations without Capture
ACHREVERSE	Used for Authorization Capture Reversal
ACHCREDIT	Used for transaction credit requests.
ACHPAYOUT	Used for ACH direct deposit payout

6.5 Authorization and Capture (ACHAUTHCAP) Request and Response

Authorization and capture occurs when a merchant processes the transaction for the entire amount therefore placing a hold on the funds and then later requests the authorized (captured) funds to be transferred to their account.

6.5.1 Authorization (ACHAUTHCAP) Request Example

```
https://api.inoviopay.com/payment/pmt_service.cfm ?
request_action=ACHAUTHCAP&request_api_version=4.14&req_username=merchant@example.com&req_
password=Gatewaypwd1&site_id=242424&request_response_format=XML&request_currency=USD&merc
h_acct_id=10001&bank_identifier=987654321&pmt_numb=01234567&cust_fname=John&cust_lname=Test
er&li_value_1=10.00&li_prod_id_1=12345&bill_addr=124%20Test%20Address&bill_addr_country=US&bill_
addr_city=Los%20Angeles&bill_addr_state=CA&bill_addr_zip=90904&xtl_ip=
```

6.5.2 Authorization (ACHAUTHCAP) Response Example (Pending Status)

```
<RESPONSE>
  <REQUEST_ACTION>ACHAUTHCAP</REQUEST_ACTION>
  <TRANS_STATUS_NAME>PENDING</TRANS_STATUS_NAME>
  <TRANS_VALUE>10 </TRANS_VALUE>
  <CURR_CODE_ALPHA>USD</CURR_CODE_ALPHA>
  <TRANS_VALUE_SETTLED>10</TRANS_VALUE_SETTLED>
  <CURR_CODE_ALPHA_SETTLED>USD</CURR_CODE_ALPHA_SETTLED>
  <TRANS_EXCH_RATE/>
  <TRANS_ID>022480</TRANS_ID>
  <CUST_ID>1234567</CUST_ID>
  <XTL_CUST_ID/>
  <PO_ID>323232</PO_ID>
  <XTL_ORDER_ID/>
  <BATCH_ID>10999</BATCH_ID>
  <PROC_NAME>ACH Processor</PROC_NAME>
  <MERCH_ACCT_ID>10001</MERCH_ACCT_ID>
  <CARD_BRAND_NAME/>
  <CARD_DETAIL/>
  <CARD_TYPE/>
  <CARD_PREPAID/>
  <CARD_BANK/>
  <CARD_BALANCE/>
  <PMT_L4>4567</PMT_L4>
  <PMT_ID>505050</PMT_ID>
  <PMT_ID_XTL/>
  <PROC_UDF01/>
  <PROC_UDF02/>
  <PROC_AUTH_RESPONSE/>
  <PROC_RETRIEVAL_NUM/>
  <PROC_REFERENCE_NUM/>
  <PROC_REDIRECT_URL/>
  <AVS_RESPONSE/>
  <CVV_RESPONSE/>
  <REQ_ID>30814003200</REQ_ID>
  <REQUEST_API_VERSION>4.14</REQUEST_API_VERSION>
  <PO_LI_ID_1>1555538</PO_LI_ID_1>
  <PO_LI_COUNT_1>1</PO_LI_COUNT_1>
  <PO_LI_AMOUNT_1>10</PO_LI_AMOUNT_1>
  <PO_LI_PROD_ID_1>12345</PO_LI_PROD_ID_1>
  <MBSHP_ID_1/>
</RESPONSE>
```

6.6 Account Validation via AUTHORIZATION without CAPTURE (ACHAUTHORIZE) Request and Response

This Request/Response is used to validate ACH parameters provided by the customer without capturing any amount. This means that there are no funds captured or transferrable to a Merchant's account.

6.6.1 Authorization (ACHAUTHORIZE) Request Example

https://api.inoviopay.com/payment/token_service.cfm?request_action=ACHAUTHORIZE&request_api_version=4.14&req_username=**merchant@example.com**&req_password=**Gatewaypwd1**&site_id=**242424**&request_response_format=**XML**&request_currency=**USD**&merch_acct_id=**10001**&bank_identifier=**987654321**&pmt_numb=**01234567**&cust_fname=**John**&cust_lname=**Tester**&li_value_1=**10.00**&li_prod_id_1=**12345**&bill_addr=**124%20Test%20Address**&bill_addr_country=**US**&bill_addr_city=**Los%20Angeles**&bill_addr_state=**CA**&bill_addr_zip=**90904**&xtl_ip=

6.6.2 Authorization (ACHAUTHORIZE) Response Example (Pending Status)

```
<RESPONSE>
  <REQUEST_ACTION>ACHAUTHCAP</REQUEST_ACTION>
  <TRANS_STATUS_NAME>PENDING</TRANS_STATUS_NAME>
  <TRANS_VALUE>10 </TRANS_VALUE>
  <CURR_CODE_ALPHA>USD</CURR_CODE_ALPHA>
  <TRANS_VALUE_SETTLED>10</TRANS_VALUE_SETTLED>
  <CURR_CODE_ALPHA_SETTLED>USD</CURR_CODE_ALPHA_SETTLED>
  <TRANS_EXCH_RATE/>
  <TRANS_ID>022480</TRANS_ID>
  <CUST_ID>1234567</CUST_ID>
  <XTL_CUST_ID/>
  <PO_ID>323232</PO_ID>
  <XTL_ORDER_ID/>
  <BATCH_ID>10999</BATCH_ID>
  <PROC_NAME>ACH Processor</PROC_NAME>
  <MERCH_ACCT_ID>10001</MERCH_ACCT_ID>
  <CARD_BRAND_NAME/>
  <CARD_DETAIL/>
  <CARD_TYPE/>
  <CARD_PREPAID/>
  <CARD_BANK/>
  <CARD_BALANCE/>
  <PMT_L4>4567</PMT_L4>
  <PMT_ID>505050</PMT_ID>
  <PMT_ID_XTL/>
  <PROC_UDF01/>
  <PROC_UDF02/>
  <PROC_AUTH_RESPONSE/>
  <PROC_RETRIEVAL_NUM/>
  <PROC_REFERENCE_NUM/>
  <PROC_REDIRECT_URL/>
  <AVS_RESPONSE/>
  <CVV_RESPONSE/>
  <REQ_ID>30814003200</REQ_ID>
  <REQUEST_API_VERSION>4.14</REQUEST_API_VERSION>
  <PO_LI_ID_1>1555538</PO_LI_ID_1>
  <PO_LI_COUNT_1>1</PO_LI_COUNT_1>
  <PO_LI_AMOUNT_1>10</PO_LI_AMOUNT_1>
  <PO_LI_PROD_ID_1>12345</PO_LI_PROD_ID_1>
  <MBSHP_ID_1/>
</RESPONSE>
```

6.6.3 Service Declined Response Example

The example above was declined by the API due to an invalid data in the PMT_NUMB field.

```
<RESPONSE>
<REQUEST_ACTION>ACHAUTHCAP</REQUEST_ACTION>
<TRANS_STATUS_NAME/>
<CUST_ID/>
<XTL_CUST_ID>c777777777</XTL_CUST_ID>
<MERCH_ACCT_ID>100</MERCH_ACCT_ID>
<CARD_BRAND_NAME/>
<PMT_L4/>
<PMT_ID/>
<PMT_ID_XTL/>
<API_RESPONSE>113</API_RESPONSE>
<API_ADVICE>Invalid Data</API_ADVICE>
<SERVICE_RESPONSE>0</SERVICE_RESPONSE>
<SERVICE_ADVICE>Declined</SERVICE_ADVICE>
<PROCESSOR_RESPONSE>0</PROCESSOR_RESPONSE>
<PROCESSOR_ADVICE> </PROCESSOR_ADVICE>
<INDUSTRY_RESPONSE>0</INDUSTRY_RESPONSE>
<INDUSTRY_ADVICE> </INDUSTRY_ADVICE>
<REF_FIELD>PMT_NUMB</REF_FIELD>
<PROC_NAME/>
<AVS_RESPONSE/>
<CVV_RESPONSE/>
<REQUEST_API_VERSION>4.14</REQUEST_API_VERSION>
</RESPONSE>
```

6.7 Reversal Request and Response

6.7.1 Reversal Request Example

```
https://api.inoviopay.com/payment/pmt_service.cfm
?request_action=ACHREVERSE&request_api_version=4.14&req_username=merchant@example.com&req_
password=Gatewaypwd1&site_id=242424&request_response_format=XML&request_ref_po_id=323232&li_
value_1=10
```

6.7.2 Reversal Response Example

```
<RESPONSE>
  <REQUEST_ACTION>ACHREVERSE</REQUEST_ACTION>
  <TRANS_STATUS_NAME>APPROVED</TRANS_STATUS_NAME>
  <TRANS_VALUE>-10</TRANS_VALUE>
  <CURR_CODE_ALPHA>USD</CURR_CODE_ALPHA>
  <TRANS_VALUE_SETTLED>-10</TRANS_VALUE_SETTLED>
  <CURR_CODE_ALPHA_SETTLED>USD</CURR_CODE_ALPHA_SETTLED>
  <TRANS_EXCH_RATE/>
  <TRANS_ID>1635078</TRANS_ID>
  <CUST_ID>1234567</CUST_ID>
  <XTL_CUST_ID/>
  <PO_ID>547848</PO_ID>
  <XTL_ORDER_ID/>
  <BATCH_ID>10999</BATCH_ID>
```

```
  <PROC_NAME>ACHProcessor</PROC_NAME>
  <MERCH_ACCT_ID>10001</MERCH_ACCT_ID>
  <CARD_BRAND_NAME/>
 <CARD_DETAIL/>
  <CARD_TYPE/>
  <CARD_PREPAID/>
  <CARD_BANK/>
  <PMT_L4>4567</PMT_L4>
  <PMT_ID>505050</PMT_ID>
  <PMT_ID_XTL/>
  <PROC_UDF01/>
  <PROC_UDF02/>
  <PROC_AUTH_RESPONSE/>
  <PROC_RETRIEVAL_NUM/>
  <PROC_REFERENCE_NUM/>
  <PROC_REDIRECT_URL/>
  <AVS_RESPONSE/>
  <CVV_RESPONSE/>
  <REQ_ID>30224993200</REQ_ID>
  <REQUEST_API_VERSION>4.14</REQUEST_API_VERSION>
  <PO_LI_ID_1>1555539</PO_LI_ID_1>
  <PO_LI_COUNT_1>1</PO_LI_COUNT_1>
  <PO_LI_AMOUNT_1>-10</PO_LI_AMOUNT_1>
  <PO_LI_PROD_ID_1>12345</PO_LI_PROD_ID_1>
  <MBSHP_ID_1/>
</RESPONSE>
```

6.8 Credit Request and Response

6.8.1 Credit Request

```
https://api.inoviopay.com/payment/pmt_service.cfm
?REQUEST_REF_PO_ID=591570&req_username=example@this.net&req_password=Password111&site_id=12345&request_response_format=JSON&request_action=ACHCREDIT&request_api_version=20&merch_acct_id=30603&CREDIT_ON_FAIL=0&client_id=15904
```

6.8.2 Credit Response

Like ACHAUTHCAP, most processors will return a status of PENDING, and the status will later be updated to APPROVED at a later date, sometimes up to ten days.

6.9 Status Change

After the ACH order has been approved, our system will update the status of the order from PENDING to APPROVED. Notification of this change can be acquired by pulling transaction data through our Order Detail Report, or by receiving postback data.

7 Boleto

Boleto (boleto bancário) is similar to the previous two transaction types in that there is still an additional step after the authorization. The difference is that the URL in the response yields the boleto, which the customer must then print and submit with payment at a qualifying location.

Brazilian addresses are composed of the following fields and must be entered in this format.

- ***The name of the logradouro (which is equivalent to an avenue or street), the number of the place in the street and the number of the floor and/or apartment (when necessary). These there fields must be explicitly entered in this order on the bill_addr field.***
- ***The neighbourhood***
- ***The city***
- ***The state (which must be informed by an abbreviation)***
- ***The postal code (numbers only).***

Example:

```
bill_addr: "Av. Paulista, 1098, 1º andar apto. 101"
Bill_addr_district: "Bela Vista"
bill_addr_city "São Paulo"
bill_addr_state "SP"
bill_addr_country: ""
bill_addr_zip : "01310000"
```

7.1 Request Parameters

Field Name	Description	Mandatory
request_action	**BOLETOAUTHCAP**	Yes
li_prod_id_x	Line Item Count - Max value is “10”	Yes
li_value_x	Line Item Transaction Amount	Yes
cust_brcpfcnpj	Customer’s CPF number	Yes
merch_acct_id	Merchant Account ID	Yes
cust_fname	Customer’s First Name	Yes
cust_lname	Customer’s Last Name	Yes
cust_birthday	Customer’s date of birth	No
bill_addr	Customer’s Billing Street Address *<street name>, <street number>, [complement – Optional}>*	Yes
bill_addr_city	Customer’s Billing City	Yes
Bill_addr_district	Customer’s Neighborhood	Yes
bill_addr_state	Customer’s Billing State	Yes
bill_addr_zip	Customer’s Billing Postal/ZIP code	Yes
bill_addr_country	Customer’s Billing Country	Yes
cust_email	Customer’s Email Address	No

request_currency	3-letter Currency Code. Ex: BRL	No

7.2 Additional Response Parameters

Field Name	Description
proc_success_url	The address to which a Customer is redirected once the authorization is complete
proc_error_url	The address to which a Customer is redirected when an authorization fails to be completed
Proc_redirect_url	The URL to display the Boleto for printing.
Proc_barcode	Contains formatted Boleto barcode data in the gateway API response

7.2.1 Boleto Request Example

```
https://api.inoviopay.com/payment/pmt_service.cfm
?cust_fname=Testy6730960&cust_lname=Tester6730960&li_count_1=1&li_prod_id_1=32805&li_
value_1=.53&cust_login=Test6730960&cust_password=Tests6730960&xtl_cust_id=Test6730960
&request_action=BOLETOAUTHCAP&merch_acct_id=128812&cust_email=Here@isatest.org&req
uest_currency=BRL&bill_addr_country=BR&cust_brcpfcnpj=629.698.570.30&bill_addr_zip=6990
0000&bill_addr=6566+Any+street&cust_phone=6566566666&bill_addr_city=Some+City&bill_add
r_state=AC&req_username=custusername&req_password=CustPassword123&site_id=11191&requ
est_response_format=JSON&request_api_version=4.14
```

7.2.2 Boleto Response Example

Below is a sample of a response to a BOLETOAUTHCAP request. The `PROC_REDIRECT_URL` is the URL to display the Boleto for printing.


```
{
: "REQUEST_ACTION":"BOLETOAUTHCAP",
: "TRANS_STATUS_NAME":"PENDING",
: "TRANS_VALUE":1.2,
: "CURR_CODE_ALPHA":"BRL",
: "TRANS_VALUE_SETTLED":1.2,
: "CURR_CODE_ALPHA_SETTLED":"BRL",
: "TRANS_EXCH_RATE":"",
: "TRANS_ID":124973612,
: "CUST_ID":45752655,
: "XTL_CUST_ID":"",
: "PO_ID":111351972,
: "XTL_ORDER_ID":"",
: "BATCH_ID":1652714,
: "PROC_NAME":"Bradesco",
: "MERCH_ACCT_ID":52618,
: "CARD_BRAND_NAME":"BOL",
: "CARD_TYPE":"",
: "CARD_PREPAID":0,
: "CARD_BANK":"",
: "CARD_BALANCE":"",
```

```
: "PMT_L4":"3456",
: "PMT_ID":49166813,
: "PMT_ID_XTL":"",
: "PROC_UDF01":"CA",
: "PROC_UDF02":"",
: "PROC_AUTH_RESPONSE":"124973612",
: "PROC_RETRIEVAL_NUM":"23792766256001113519572003852208675100000000120",
: "PROC_BARCODE":"23792766256001113519572003852208675100000000120",
: "PROC_REFERENCE_NUM":"35cc6ac8-5ef7-4a49-b8dd-72609e2aa627",
: "PROC_REDIRECT_URL": "https://transaction.pagador.com.br/post/pagador/reenvia.asp/35cc6ac8-
5ef7-4a49-b8dd-72609e2aa627",
: "AVS_RESPONSE":"",
: "CVV_RESPONSE":"",
: "REQ_ID":"30814113221",
  "REQUEST_API_VERSION":"4.14",
: "PO_LI_ID_1":"60915685",
: "PO_LI_COUNT_1":"1",
: "PO_LI_AMOUNT_1":"1.2",
: "PO_LI_PROD_ID_1":"20923",
: "MBSHP_ID_1":""
}
```


8 Pix Payment

Pix is a payment method for instant direct bank transfers in Brazil. Pix is owned and regulated by Central Bank (Banco Central) and operated by Brazilian banks, digital accounts, and wallets.

For online shoppers using Pix, the funds are sent directly from the shopper bank account and then transferred to the merchant's bank account instantly.

8.1 PIX Request Parameters

Field Name	Description	Mandatory
request_action	**PIXSALE**	Yes
li_prod_id_x	Line Item Count - Max value is "10"	Yes
li_value_x	Line Item Transaction Amount	Yes
cust_brcpfcnpj	Customer's CPF number	Yes
merch_acct_id	Merchant Account ID	Yes
cust_fname	Customer's First Name	Yes
cust_lname	Customer's Last Name	Yes
cust_birthday	Customer date of birth	No

bill_addr	Customer Billing Street Address	No
bill_addr_city	Customer's Billing City	No
Bill_addr_district	Customer's Neighborhood	No
bill_addr_state	Customer's Billing State	No
bill_addr_zip	Customer's Billing Postal/ZIP code	No
bill_addr_country	Customer's Billing Country	No
cust_email	Customer's Email Address	No
request_currency	3-letter Currency Code. Ex: BRL	No

8.2 Additional Response Parameters

Field Name	Description
Proc_redirect_url	The address to which a Customer is redirected once the QR Code has been generated
proc_retrieval_num	Processor tracking number (*pedido.numero*)
Proc_reference_num	Contains Pix transaction verification number (*pix.token*)

8.3 Pix Request Example

```
https://api.inoviopay.com/payment/pmt_service.cfm
?request_action=PIXSALE&li_count_1=1&li_prod_id_1=32221&li_value_1=1.00&bill_addr_zip=014
18102&xtl_udf01=Test1&BILL_ADDR_CITY=Sao+Paulo&BILL_ADDR=277+Domingos+Sergio+dos+A
njos&BILL_ADDR_STATE=SP&XTL_IP=000.000.001&BILL_ADDR_DISTRICT=Pirituba&CUST_B
RCPFCNPJ= 483.649.730-29
&req_username=pixuser%40Testpay.com&req_password=Password12345555&site_id=22619&req
uest_response_format=JSON&request_api_version=4.14&request_currency=BRL&merch_acct_id
=31219&cust_email=test%40test.com&cust_fname=Pessoa&cust_lname=de+Testing
```

8.4 Pix Response Example

Below is a sample response for a PIXSALE request. The `PROC_REDIRECT_URL` is the URL to display the QR Code for scanning and further payment.


```
{
: "REQUEST_ACTION":"PIXSALE",
: "REQ_ID":"6960183",
: "TRANS_STATUS_NAME":"PENDING",
: "TRANS_VALUE":1,
: "TRANS_VALUE_SETTLED":1,
: "CURR_CODE_ALPHA_SETTLED":"BRL",
: "TRANS_EXCH_RATE":"",
: "TRANS_ID":362000021,
```

```
:  "CUST_ID":3546515,
:  "XTL_CUST_ID":"",
:  "CUST_BRCPFCNPJ":" 483.649.730-29",
:  "PO_ID":35739669,
:  "XTL_ORDER_ID":"",
:  "BATCH_ID":30323,
:  "PROC_NAME":"BRADESCO PIX",
:  "MERCH_ACCT_ID":11119,
:  "CARD_BRAND_NAME":"PIX",
:  "PMT_L4":"2488",
:  "PMT_ID":331179,
:  "PMT_ID_XTL":"",
:  "PMT_AAU_UPDATE_DT":"",
:  "PMT_AAU_UPDATE_DESC":"",
:  "PROC_UDF01":"",
:  "PROC_UDF02":"",
:  "PROC_AUTH_RESPONSE":"",
:  "PROC_RETRIEVAL_NUM":"3212739",
:
"PROC_REFERENCE_NUM":"abcdhsU2xuUFZnNmRXa25CUlM0QWJ4OEhMc2htb1o2dDNsVStDcacgsjd
NOHlWTEtZZUYzxbstsZkR2R3FkUUFoNA==",
:  "PROC_BARCODE":"",
:  "PROC_REDIRECT_URL":"
https://meiosdepagamentobradesco.com.br/apipix/formulario?token=abcdesdW5SaVVFcFVGeFFGMnBF
SGFvakdzWGFiaktmMkI3T1JkkkpvSDN4NmZWUEdTK3ovYUErMERmMmVLMStIVVdPRFdaWUdxRGxy
z==",
:  "AVS_RESPONSE":"",
:  "CVV_RESPONSE":"",
:  "REQ_ID":"30819993221",
   "REQUEST_API_VERSION":"4.14",
:  "P3DS_VENDOR":"",
:  "P3DS_RESPONSE":"",
:  "PO_LI_ID_1":"37943",
:  "PO_LI_COUNT_1":"1",
:  "PO_LI_AMOUNT_1":"1",
:  "PO_LI_PROD_ID_1":"32221",
:  "MBSHP_ID_1":""
}
```


9 Credilink

Credilink validates the Brazilian identification numbers, CPF (individuals) or CNPJ (businesses) for use on different business rules. Credilink Service is only available in the Brazilian market. For more information on technical usage, please contact your gateway support representative.

9.1 How it Works

Merchant sends transaction data, including the CPF. This can be sent with the other optional parameters like phone number, date of birth and customer name. Credilink will check the CPF and attempt to confirm the validity of the CPF/CNPJ against requested services as supplied to the gateway. If there is no match for the CPF/CNPJ, the user is a minor where restricted or if there is a death record associated with the CPF, then a decline response will be returned and other adverse actions may be enforced for the declined CPF.

- Credilink Identity check will be applicable on all transaction types (Credit Card, Pix and Boleto)
- The service will validate if the CPF belongs to a minor or not. (No date of birth will be returned)
    - If the CPF is for a minor (person of less than 18 YO as per the date of check), then a Response indicating the Customer as a minor will be returned
    - If the CPF is not for a minor, then additional checks will be performed
- The Service will also validate if the CPF is associated with a death record.
    - If the CPF is associated with a death record, a response indicating a death record will be returned
    - If there is a death record associated with the CPF, then the transaction will be declined and CPF is blacklisted.
- The Service also checks for CPF validity, if CPF is reported as invalid, then a response is returned indicating invalid cpf
- The service will check if the CPF is valid but canceled, if the CPF is canceled, then a response is returned indicating invalid cpf.

9.2 How to Use Credilink

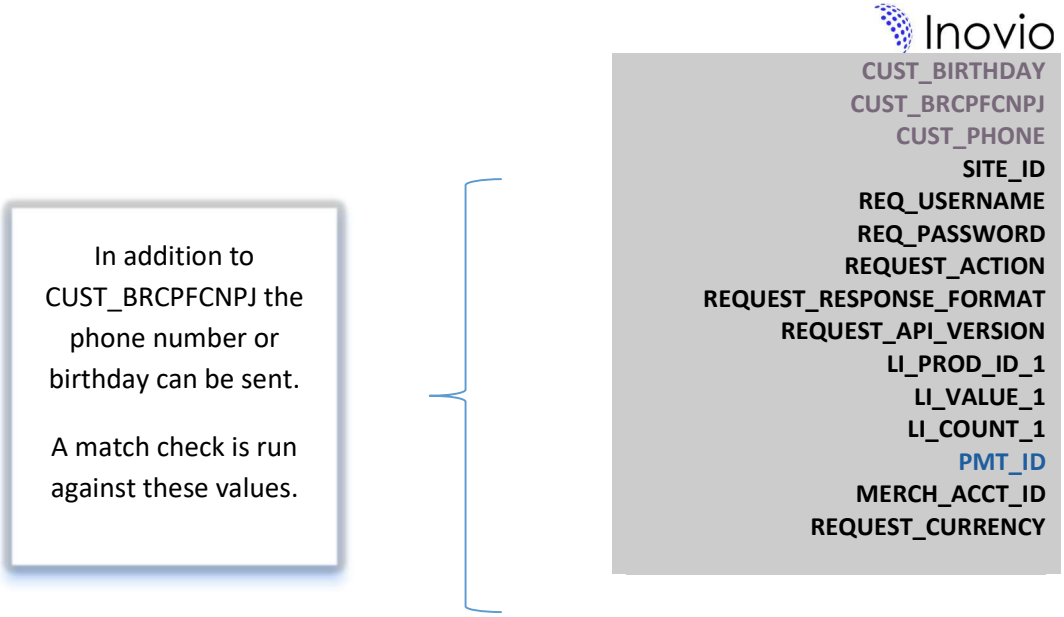
- Submit a request to Inovio to configure and activate Credilink on a merchant account that'll use the credilink services
- Pass in the CUST_BRCPFCNPJ parameter in every transaction
- Receive response:

Service_Response	Description
700	Scrub Decline
706	Failed Age Validation
707	Invalid CPF

9.3 Additional Parameters

Field Name	Condition	Description
CUST_BRCPFCNPJ	Required	Customer's cpf/cnpj
CUST_BIRTHDAY	Optional	Customer's Date of Birth
CUST_PHONE	Optional	Customer's Phone number
CUST_FNAME	Optional	Customer's First Name
CUST_LNAME	Optional	Customer's Last Name

Transaction Fields



10 Order Insight

Order Insight is a platform that allows merchants to share transaction information with issuers in real time to resolve disputes and prevent chargebacks. It's part of Visa's initiative to modernize the dispute management process and is integrated into Visa's issuer dispute management platform, Visa Resolve Online (VROL). Verifi is a payment protection and management company that helps merchants, issuers, and acquirers reduce financial loss and fraud.

- Order Insight allows Merchants to receive real-time alerts when a transaction dispute is initiated by a customer. The Order Insight service will share and receive detailed transaction data on the customer, the product or service purchased, and the payment method used.
- Compelling Evidence (CE) is an additional tool of Order Insight that helps merchants use a cardholder's purchase history to prove that a disputed transaction is legitimate. This takes place over 2-5 separate API requests to confirm the consistency of the recurring purchase data.

10.1 Additional Request Parameters

The below request parameters are needed to fully unlock the benefits of Verifi's Order Insight product. All transaction linking through API calls is completed on behalf of the merchant's account by Inovio. These are optional fields within the Inovio Gateway API and will not stop a transaction even if Order Insight is enabled on the Merchant Account. When an Order Insight request comes in, Inovio will respond with all transactional information given by the merchant at the time of the original transaction.

Order Insight requires a history of 120 days of the product parameters described below. CE requires a history of 365 days of the customer and device parameters described below.

For more information on technical usage, please contact your gateway support representative to obtain a copy of the form to be filled out to expedite the historical update of transactional information and use of Order Insight products.

Field Name	Description	Data Type	Char Limit
XTL_CUST_ID	Merchant's Customer ID	Alphanumeric and special characters	max of 24
PROD_DESC_XTL	Detailed description of the product (merchandise or service) purchased	Alphanumeric and special characters	up to 1,000
DEVICE_ID_XTL	Device ID of the device used to submit the order. Examples are IMEI or MEID values from the device. Data cannot be hashed.	Alphanumeric and special characters	15-32
DEVICE_FINGERPRINT_XTL	Device fingerprint information is generated by a third-party service provider or the Merchant's own algorithm to combine browser or device attributes to form a unique fingerprint to identify the device. Data can be hashed.	Alphanumeric and special characters	20-45

10.2 Reporting

10.2.1 Order Insight Lookup Statuses

As an Order Insight lookup ages, postbacks will be sent to show the progression throughout the transaction's lifetime. A lookup can have many statuses depending on the outcome of the API calls between Inovio and Verifi over the transaction's chargeback window.

Status	Description
Responded	Inovio has received the API call from Verifi and responded with the data requested within SLA. This is followed by a "ORDER_INSIGHT_LOOKUP_EVENT" postback.
Failed (visible in Portal and Order Detail API)	Inovio never received the API request from Verifi. Lookup event was discovered during the SFTP catch up process. We will fight to prevent this charge from Verifi.
Deflection	The lookup request did not lead to a chargeback. This is followed by a "ORDER_INSIGHT_SETTLED_EVENT" postback.
Negation	The lookup request led to a chargeback. This is followed by a "ORDER_INSIGHT_NEGATION_EVENT" postback.
Reversal	The lookup request that was previously led to a deflected dispute was reversed. This is followed by a "ORDER_INSIGHT_REVERSAL_EVENT" postback.

11 Transaction Modes

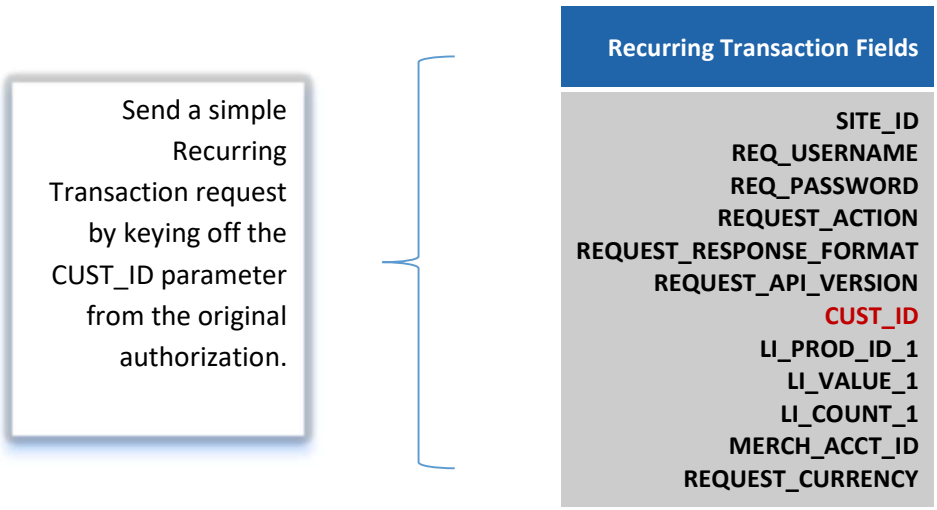
11.1 Recurring

Merchants may sometimes need to charge their customers periodically using a payment account that a customer has previously used. In this case, the succeeding transaction after the original

authorization is called a *recurring* transaction. In recurring transaction mode, the merchant is not required to re-send the full payment information. When a client intends to use this functionality, the first customer initiated transaction should be flagged as request_rebill=2. This makes the processor aware that this is the initial transaction of an upcoming recurring card on file transaction or set of transactions. See Appendix K for more details.

11.1.1 Minimum required fields for sending a simple recurring transaction request using CUST_ID:

A recurring transaction request can be made by sending the cardholder's Customer ID or *cust_id* in the transaction request. The Payment Service will charge the most recent payment account or credit card that was used. The customer's most recently used address record will be used in the transaction request. Non-USD rebill requests should include the REQUEST_CURRENCY with the non-USD currency 3-letter code.



11.1.2 Required fields for specifying which payment account to charge with the recurring transaction request using PMT_L4 parameter:

Some customers may have more than one payment account in the system. In order to specify which payment account to charge, merchants may send the last 4 digits of the payment account or credit card number in the *pmt_l4* parameter.

Sending the last 4 digits of the payment account will let the Payment Service know which account to charge for the recurring payment.

*As an option, merchants may also send a new Credit Card Expiration Date in the request.

Recurring Transaction Fields
SITE_ID
REQ_USERNAME
REQ_PASSWORD
REQUEST_ACTION
REQUEST_RESPONSE_FORMAT
REQUEST_API_VERSION
CUST_ID
LI_PROD_ID_1
LI_VALUE_1
LI_COUNT_1
PMT_L4
*PMT_EXPIRY
MERCH_ACCT_ID
REQUEST_CURRENCY

11.1.3 Required fields for specifying which payment account to charge with the recurring transaction request using PMT_ID parameter:

Alternatively, the merchant may use Payment ID (PMT_ID) instead of PMT_L4, in specifying which payment record to use in the transaction request. The Payment Service will return an error if it cannot find the payment record.

Sending the Payment ID of the payment account will let the Payment Service know which account to charge for the recurring payment.

.

Recurring Transaction Fields
SITE_ID
REQ_USERNAME
REQ_PASSWORD
REQUEST_ACTION
REQUEST_RESPONSE_FORMAT
REQUEST_API_VERSION
CUST_ID
LI_PROD_ID_1
LI_VALUE_1
LI_COUNT_1
PMT_ID
MERCH_ACCT_ID
REQUEST_CURRENCY

11.1.4 Required fields for specifying which payment account to charge with the recurring transaction request using PMT_ID_XTL parameter:

Merchants may also use their External Payment ID (PMT_ID_XTL) instead of PMT_ID or PMT_L4.

Sending the External Payment ID works the same way as sending the PMT_L4 or PMT_ID. The service will return an error if it cannot find the customer's payment record.

Recurring Transaction Fields
SITE_ID
REQ_USERNAME
REQ_PASSWORD
REQUEST_ACTION
REQUEST_RESPONSE_FORMAT
REQUEST_API_VERSION
CUST_ID
LI_PROD_ID_1
LI_VALUE_1
LI_COUNT_1
PMT_ID_XTL
MERCH_ACCT_ID
REQUEST_CURRENCY

11.1.5 Required parameters for sending new billing information with the recurring transaction request:

New billing information including the address fields may be sent with the Recurring Transaction request.

Card expiration date and **CVV2/CVC2** are required when sending the full credit card number in the transaction.

The ***Address Fields** are optional. The Payment Service will use the most recent address information on file if new address information were not included in the request.

Recurring Transaction Fields
SITE_ID
REQ_USERNAME
REQ_PASSWORD
REQUEST_ACTION
REQUEST_RESPONSE_FORMAT
REQUEST_API_VERSION
CUST_ID
LI_PROD_ID_1
LI_VALUE_1
LI_COUNT_1
PMT_NUMB
PMT_EXPIRY
PMT_KEY
PMT_ID_XTL
*BILL_ADDR
*BILL_ADDR_CITY
*BILL_ADDR_STATE
*BILL_ADDR_COUNTRY
*BILL_ADDR_ZIP
MERCH_ACCT_ID
REQUEST_CURRENCY

11.1.6 Parameters for sending Renewal Transactions (Rebills):

NOTE: This is for merchant-managed Membership products and should not be confused with Renewals managed by the gateway.

To specify that a request is a Renewal or Rebill, set the REQUEST_REBILL parameter to "1".

Merchants are encouraged to always set the REQUEST_REBILL parameter for Renewal Transactions as it is important in generating customer and subscription reports.

Non-USD rebill request should include the REQUEST_CURRENCY with the non-USD currency 3-letter code.

REQUEST_REBILL = 1 tells the API to record the transaction as a Rebill/Renewal transaction.

The Address and Payment Fields are optional. CUST_ID maybe optional; contact your gateway support representative for details.

Recurring Transaction Fields
SITE_ID
REQ_USERNAME
REQ_PASSWORD
REQUEST_ACTION
REQUEST_RESPONSE_FORMAT
REQUEST_API_VERSION
CUST_ID
LI_PROD_ID_1
LI_VALUE_1
LI_COUNT_1
PMT_NUMB
PMT_EXPIRY
PMT_ID_XTL
BILL_ADDR
BILL_ADDR_CITY
BILL_ADDR_STATE
BILL_ADDR_COUNTRY
BILL_ADDR_ZIP
REQUEST_REBILL = 1
MERCH_ACCT_ID
REQUEST_CURRENCY

11.1.6.1 Example Rebill Request

Merchants can send the CUST_ID to charge an existing customer in the system, or send the complete payment and customer information in the request, if CUST_ID is not available.

```
Pmt_service.cfm?req_password=TestPassword1&site_id=0&request_response_format=JSON&req
_username=test%40example.com&li_value_1=25.75&li_prod_id_1=1001&request_currency=EUR&
xtl_order_id=test&cust_id=1501&pmt_l4=0026&request_action=CCAUTHCAP&merch_acct_id=100
&request_rebill=1
```

11.2 Memberships

The payment system supports website membership management, commonly known as "subscriptions". Product IDs (li_prod_id_1) should be created before implementing this type of transaction, which can be created using the gateway's Product Service. For more information on technical usage, please contact your gateway support representative.

Some processors may require customer billing address data.

11.2.1 Suggested fields for sending a membership transaction (new customer):

Field Name	Description	Data Type
REQUEST_ACTION	Service Request Action (Send “CCAUTHCAP”).	Service Request Types
REQ_USERNAME	Service Request Username	Alphanumeric
REQ_PASSWORD	Service Request Password	Alphanumeric and Special Characters
REQUEST_RESPONSE_FORMAT	Service Response Format	Accepted values: “XML”, “PIPES” and ”JSON”
REQUEST_API_VERSION	Payment Service API Version	Numeric
SITE_ID	Merchant’s Website ID	Numeric
CUST_FNAME	Customer First Name	Alphanumeric and Special Characters
CUST_LNAME	Customer Last Name	Alphanumeric and Special Characters
CUST_EMAIL	Customer Email Address	Alphanumeric
LI_COUNT_1	Line Item Count Max value is “99”.	Numeric
LI_PROD_ID_1	Line Item Product ID 1 (product type should be set to membership)	Numeric
LI_VALUE_1	Line Item Transaction Amount 1	Numeric
BILL_ADDR	Customer Billing Street Address	Alphanumeric
BILL_ADDR_CITY	Customer Billing City	Alphanumeric
BILL_ADDR_STATE	Customer Billing State	2-letter State or Territory Code
BILL_ADDR_ZIP	Customer Billing Postal/ZIP code	Alphanumeric
BILL_ADDR_COUNTRY	Customer Billing Country	2-letter Country Code ISO 3166-1 alpha-2
PMT_NUMB	Credit Card Number	Numeric
PMT_KEY	Credit Card CVV2 or CVC2 Code	Numeric (4)
PMT_EXPIRY	Credit Card Expiration Date	Numeric MMYYYY Example: “122014”

CUST_LOGIN	Customer Login or User Name	Alphanumeric and Special Characters
CUST_PASSWORD	Cardholder's Password Password must be at least 10 characters with 1 number, lower case and upper case letter. NOTE: Contact Client Services to request change of password requirements.	Alphanumeric
MERCH_ACCT_ID	Merchant Account ID	Numeric
REQUEST_CURRENCY	3-letter Currency Code	Example: "USD"
REQUEST_AFF_ID	External Affiliate ID	Alphanumeric
PMT_ID_XTL	External Payment/Credit Card ID	Alphanumeric

11.2.2 Parameters to create Subscription directly from the Gateway API

For more information on technical usage, please contact your gateway support representative about creating subscriptions directly from the Gateway API.

Field Name	Description
PROD_NAME	Product name
PROD_TYPE	1 - Membership Cancels 2 - Membership Renews
PROD_REBILL_METRIC	M Month D Day Y Year
PROD_REBILL_PERIOD	7, 30, 90 etc

11.2.2.1 Example Request creating Subscription directly from Gateway API

```
https://api.inoviopay.com/payment/pmt_service.cfm
?request_action=CCAUTHCAP&pmt_numb=4000000000000002&pmt_expiry=122025&li_value_1=1.00&bill_addr_state=CA&bill_addr_zip=63146&pmt_key=123&xtl_udf01=Test01&prod_name=The+Name&prod_type=2&prod_rebill_metric=D&prod_rebill_period=1&req_username=api@fauxcom.net&req_password=Password9993&site_id=23803&request_response_format=JSON&request_api_version=4.14&request_currency=USD&argus_debug=0
```

11.2.2.2 Example Response creating Subscription directly from Gateway API


```
{
  "CUST_ID": 110750,
  "REQUEST_API_VERSION": "4.14",
  "CURR_CODE_ALPHA_SETTLED": "USD",
  "CARD_PREPAID": 0,
  "XTL_CUST_ID": "",
```

```
  "CARD_TYPE": "",
  "CARD_DETAIL": "",
  "P3DS_RESPONSE": "",
  "PMT_ID": 525551,
  "BATCH_ID": 124104,
  "TRANS_VALUE_SETTLED": 1,
  "TRANS_ID": 1797217,
  "CURR_CODE_ALPHA": "USD",
  "PMT_AAU_UPDATE_DT": "",
  "PROC_REFERENCE_NUM": "TEST828185469",
  "MERCH_ACCT_ID": 123403,
  "REQUEST_ACTION": "CCAUTHCAP",
  "PO_ID": 988074,
  "PROC_NAME": "Test Processor",
  "PMT_L4": "0002",
  "MBSHP_ID_1": "73102",
  "TRANS_STATUS_NAME": "APPROVED",
  "XTL_ORDER_ID": "",
  "TRANS_EXCH_RATE": "",
  "P3DS_VENDOR": "",
  "AVS_RESPONSE": "M",
  "REQ_ID": "15104062",
  "PMT_ID_XTL": "",
  "PO_LI_ID_1": "1829990",
  "PMT_AAU_UPDATE_DESC": "",
  "PROC_UDF01": "",
  "PROC_RETRIEVAL_NUM": "29259606-88FD-4545-808FBD6936846650",
  "CARD_BALANCE": "",
  "CARD_BRAND_NAME": "Visa",
  "CARD_CLASS": "Consusmer Credit",
  "CARD_DETAIL": "Credit",
  "CARD_COUNTRY": "USA",
  "TRANS_VALUE": 1,
  "PROC_UDF02": "",
  "PO_LI_COUNT_1": 1,
  "PROC_REDIRECT_URL": "",
  "CARD_BANK": "",
  "PROC_AUTH_RESPONSE": "TEST31436",
  "CVV_RESPONSE": "M",
  "PO_LI_AMOUNT_1": "1.00",
  "PO_LI_PROD_ID_1": 127617
}
```


11.2.3 Unique Customer Login Check

For new customers, Payment Service will decline requests if the customer login (CUST_LOGIN) information sent in the request already exists within the same website (SITE_ID). The gateway will return Service Response “695” on failed Unique Customer Login Check.

11.2.3.1 Example Declined Response: Username Unavailable


```
<RESPONSE>
  <REQUEST_ACTION>CCAUTHCAP</REQUEST_ACTION>
  <TRANS_STATUS_NAME/>
  <TRANS_VALUE/>
```

```
  <TRANS_ID/>
  <CUST_ID/>
  <XTL_CUST_ID/>
  <MERCH_ACCT_ID>100</MERCH_ACCT_ID>
  <CARD_BRAND_NAME/>
  <PMT_L4/>
  <API_RESPONSE>0</API_RESPONSE>
  <API_ADVICE></API_ADVICE>
  <SERVICE_RESPONSE>695</SERVICE_RESPONSE>
  <SERVICE_ADVICE>Site Username Unavailable</SERVICE_ADVICE>
  <PROCESSOR_RESPONSE>0</PROCESSOR_RESPONSE>
  <PROCESSOR_ADVICE></PROCESSOR_ADVICE>
  <INDUSTRY_RESPONSE>0</INDUSTRY_RESPONSE>
  <INDUSTRY_ADVICE></INDUSTRY_ADVICE>
  <REF_FIELD/>
  <PROC_NAME/>
  <AVS_RESPONSE/>
  <CVV_RESPONSE/>
  <REQUEST_API_VERSION>4.14</REQUEST_API_VERSION>
</RESPONSE>
```

11.2.4 Membership Cancelation Requests (SUB_CANCEL)

Merchants may send cancelation requests using the Payment Service by sending request_action=SUB_CANCEL in the gateway request. Below are the minimum required parameters for cancelation requests:

11.2.4.1 SUB_CANCEL Request Parameters

Field Name	Description	Data Type
REQ_USERNAME	Service Request Username	Alphanumeric
REQ_PASSWORD	Service Request Password	Alphanumeric and Special Characters
REQUEST_RESPONSE_FORMAT	Service Response Format	Accepted values: “XML” and ”JSON”
REQUEST_API_VERSION	Payment Service API Version (API Version needs to be 2 or higher.)	Numeric
SITE_ID	Merchant’s Website ID	Numeric
REQUEST_ACTION	Service Request Action: send “**SUB_CANCEL**”	
REQUEST_REF_MBSHP_ID	Referring Membership ID. Note: MBSHP_ID is returned on successful membership authorization requests.	Numeric

SUB_CANCEL_TYPE	Types: **"1"= CANCEL NOW** "1" will set the membership cancelation date to today's date. **"2" = CANCEL ON NEXT UPCOMING REBILL DATE** Send "2" to request membership cancelation on the rebill/renewal date.	

11.2.4.2 SUB_CANCEL Gateway Request Example

```
request_ref_mbshp_id=9149&req_password=Testpass1234&site_id=1234&sub_cancel_type=1&request_api_version=4.14&req_username=testmerchant@example.com&cust_id=291087&request_action=SUB_CANCEL
```

11.2.4.3 SUB_CANCEL Response Parameters

Field Name	Description
REQUEST_ACTION	Request Action Type: "SUB_CANCEL"
API_RESPONSE	API Response Code
API_ADVICE	API Response Advice
SERVICE_RESPONSE	Service Response Code
SERVICE_ADVICE	Service Response Advice
REF_FIELD	Reference Field (this will contain the field in question in case of data validation errors)
CUST_ID	Customer ID
MBSHP_ID	Membership ID
MBSHP_CANCEL_TS_UTC	Membership Cancelation Date in UTC
MBSHP_REBILL_TS_UTC	Membership Rebill Date in UTC

11.2.4.4 SUB_CANCEL Gateway Response Example


```
{
  "REQUEST_ACTION": "SUB_CANCEL",
  "API_RESPONSE": "0",
  "API_ADVICE": " ",
  "SERVICE_RESPONSE": 0,
  "SERVICE_ADVICE": " ",
  "REF_FIELD": "",
  "CUST_ID": 8868,
  "MBSHP_ID": 817,
  "MBSHP_CANCEL_TS_UTC": "October, 30 2025 19:17:59",
  "MBSHP_REBILL_TS_UTC": ""
}
```

11.2.5 Membership Product Update Requests (Upgrade/downgrade Product)

Merchants may send product update requests using the Payment Service by sending request_action=SUB_UPDATE in the gateway request. Note that if the update product is a membership, a new membership record (with the same customer information) is created when the customer's current membership product has been successfully changed. The customer is charged for the update product on the next upcoming rebill date of the customer's current membership. The customer's current membership will be canceled at the end of the subscription date.

Below are the minimum required parameters for SUB_UPDATE requests:

11.2.5.1 SUB_UPDATE Request Parameters

Field Name	Description	Data Type
REQ_USERNAME	Service Request Username	Alphanumeric
REQ_PASSWORD	Service Request Password	Alphanumeric and Special Characters
REQUEST_RESPONSE_FORMAT	Service Response Format	Accepted values: "XML" and "JSON"
REQUEST_API_VERSION	Payment Service API Version (4.14)	Numeric
SITE_ID	Merchant's Website ID	Numeric
REQUEST_ACTION	Service Request Action: send "**SUB_UPDATE**"	
REQUEST_REF_MBSHP_ID	Referring Membership ID.	Numeric
SUB_UPDATE_PROD_ID	Membership will be updated to the Product ID sent in this field.	Numeric
SUB_UPDATE_PMT_ID	Used to Update Card used for Membership (tied to specific Membership)	
SUB_UPDATE_TYPE	Immediately change a customer's membership/subscription from Product A to Product B and run a (Payment Type) AUTHCAP for Product B.	Optional Acceptable value "1" Response will be that of a *AUTHCAP

11.2.5.2 SUB_UPDATE Response Parameters

Field Name	Description
REQUEST_ACTION	Request Action Type: "SUB_UPDATE"
API_RESPONSE	API Response Code

API_ADVICE	API Response Advice
SERVICE_RESPONSE	Service Response Code
SERVICE_ADVICE	Service Response Advice
REF_FIELD	Reference Field (this will contain the field in question in case of data validation errors)
CUST_ID	Customer ID
MBSHP_ID	Membership ID
MBSHP_CANCEL_TS_UTC	Membership Cancelation Date in UTC
MBSHP_REBILL_TS_UTC	Membership Rebill Date in UTC
SUB_VALUE_REMAINDER	Amount of the remaining value of membership/subscription A. - Applicable when SUB_UPDATE_TYPE is in the request.
PREV_PO_ID	Previous Order ID to be used when sending adjustment requests against authorizations (i.e. Delayed Capture, Reversal and Credit requests). - Applicable when SUB_UPDATE_TYPE is in the request.
SUB_UPDATE	Confirmation that the immediate membership/subscription change has taken place. - Applicable when SUB_UPDATE_TYPE is in the request.

*11.2.5.3 SUB_UPDATE Gateway Request Example (*SUB_UPDATE_PMT_ID*)*

Merchants may update a Payment Card (PMT_ID) through the gateway services for an existing Membership without having to cancel the Membership by sending request_action=SUB_UPDATE in the gateway request together with **SUB_UPDATE_PMT_ID** Parameter with the replacement PMT_ID. If the customer is adding a new card that doesn't exist in our system for that customer, clients shall run a **CCAUTHORIZE** for zero amount ($0.00) to validate the card and create a PMT_ID that'll be used with SUB_UPDATE_PMT_ID.

```
/pmt_service.cfm?request_action=SUB_UPDATE&req_username=testmerchant@example.com&req_password=TestPass1234&request_ref_mbshp_id=1234&site_id=1234&SUB_UPDATE_PROD_ID=78890&request_api_version=4.14&SUB_UPDATE_PMT_ID=546789&cust_id=291061
```

11.2.5.4 *SUB_UPDATE Gateway Response Example (*SUB_UPDATE_PMT_ID*)*


```
{
 "REQUEST_ACTION": "SUB_UPDATE",
 "API_RESPONSE": "0",
 "API_ADVICE": " ",
 "SERVICE_RESPONSE": 102,
 "SERVICE_ADVICE": "Membership Updated",
 "REF_FIELD": "",
 "CUST_ID": 7538691,
 "MBSHP_ID": 78245,
```

```
 "MBSHP_CANCEL_TS_UTC": "June, 28 2024 22:00:59",
 "MBSHP_REBILL_TS_UTC": ""
}
```


12 Response Fields

12.1 Response Formats

- Extensive Mark-up Language (XML)
- JavaScript Object Notation (JSON)

12.2 Successful Transaction Response Fields

These are the fields returned to the merchant when a transaction has been approved by the Payment Service.

Field Name	Description
REQUEST_ACTION	This will return the Service Request Action the merchant sent in the transaction request.
TRANS_STATUS_NAME	Transaction Status
TRANS_VALUE	Total requested transaction amount for all line items.
CURR_CODE_ALPHA	Requested Currency 3-letter Code
TRANS_VALUE_SETTLED	Transaction Settled Amount (after conversion to settled currency).
CURR_CODE_ALPHA_SETTLED	Settled Currency 3-letter Code
TRANS_EXCH_RATE	Currency Exchange Rate
TRANS_ID	Transaction ID
CUST_ID	Customer ID
XTL_CUST_ID	Merchant's Customer ID
PO_ID	Purchase order ID
XTL_ PO_ID	Merchant's Order ID
BATCH_ID	Settlement Batch ID
PROC_NAME	Merchant Processor Name (Example: "EPX")
MERCH_ACCT_ID	Merchant Bank's Account ID

CARD_BRAND_NAME	Credit Card Network/Brand Name
CARD_DETAIL	Credit or Debit Card
CARD_TYPE	Credit Card Type
CARD_CLASS	Categorizes the BIN as a Business, Corporate, Purchase, or Consumer card
CARD_COUNTRY	Issuer Bank country for the BIN
CARD_PREPAID	Indicates that the credit card is a prepaid card if value returned is “1”.
CARD_BANK	Credit Card Issuing Bank Name
CARD_BALANCE	Prepaid card balance (this is a processor-specific feature). This field will return the card’s available balance.
PMT_L4	Payment account or credit card’s last 4 digits.
PMT_ID	Payment Unique Identifier
PMT_ID_XTL	External Unique Identifier
PROC_UDF01	Processor User Defined Field 1
PROC_UDF02	Processor User Defined Field 2
PROC_AUTH_RESPONSE	Processor Authorization Response Code
PROC_RETRIEVAL_NUM	Processor Retrieval Number or GUID
PROC_REFERENCE_NUM	Processor Reference Number
PROC_REDIRECT_URL	URL where customers are redirected to for external verification (i.e. 3D Secure page)
AVS_RESPONSE	Address Verification Service Response Code
CVV_RESPONSE	Card Verification Value Response Code
REQUEST_API_VERSION	Payment Service API Version
PO_LI_ID_X	Purchase Order Line Item ID
PO_LI_COUNT_X	Purchase Order Line Item Count
PO_LI_AMOUNT_X	Purchase Order Line Item Total Amount

PO_LI_PROD_ID_X	Purchase Order Line Item Product ID
MBSHP_ID	Membership ID (returned on membership transactions
TRANS_NTOKEN_USED	Used to indicate on whether a Network Token was used or a PAN was used. (It will be set to "1" if a scheme token was used or set to "0" otherwise)
CARD_BRAND_TRANSID	This is the card scheme transaction id for the current transaction

Important Note

If there is more than one line item in the transaction request you sent, the API will return the succeeding line item response fields for each line item that was created (PO_LI_ID_X, PO_LI_COUNT_X, PO_LI_AMOUNT_X, and PO_LI_PROD_ID_X ("x" is to indicate a dynamic number depending on the number of line items sent in the request)). For example, ***sending two line items*** in your gateway request will get a response like this:

```
< <RESPONSE>

<PO_LI_ID_1>123455</PO_LI_ID_1>
<PO_LI_COUNT_1>1</PO_LI_COUNT_1>
<PO_LI_AMOUNT_1>19.95</PO_LI_AMOUNT_1>
<PO_LI_PROD_ID_1>1001</PO_LI_PROD_ID_1>
<PO_LI_ID_2>123456</PO_LI_ID_2>
<PO_LI_COUNT_2>1</PO_LI_COUNT_2>
<PO_LI_AMOUNT_2>10.5</PO_LI_AMOUNT_2>
<PO_LI_PROD_ID_2>2001</PO_LI_PROD_ID_2>
</RESPONSE>
```

LINE ITEM FIELDS

12.3 Declined Transaction Response Fields

These are the fields returned to the merchant when the transaction request has been declined on the Merchant Account Processor (bank) level:

Field Name	Description
REQUEST_ACTION	This will return the Service Request Action the merchant sent in the transaction request.
TRANS_STATUS_NAME	Transaction Status
TRANS_VALUE	Total transaction amount
TRANS_ID	Unique transaction ID
CUST_ID	Customer ID
XTL_CUST_ID	Merchant's Customer ID.
MERCH_ACCT_ID	[WL] Account ID
CARD_BRAND_NAME	Bank account or credit card type (Example: "MasterCard").
PMT_L4	Payment account or credit card's last 4 digits.

API_RESPONSE	Payment Service API Response Code. This is the higher level response information used for authentication and checking Service availability.
API_ADVICE	Payment Service API Response details.
SERVICE_RESPONSE	Service Response Code. The information in this field is for all transaction related requests (example: authorization requests and scrub-only transaction requests.)
SERVICE_ADVICE	Service Response details
PROCESSOR_RESPONSE	Merchant Account Processor Response Code
PROCESSOR_ADVICE	Merchant Account Service Response details
INDUSTRY_RESPONSE	Issuing Bank's Response Code
INDUSTRY_ADVICE	Issuing Bank's Response details
REF_FIELD	Reference field (useful when getting "missing required parameter" messages).
PROC_NAME	Merchant Processor Name (Example: Payvision)
AVS_RESPONSE	Address Verification Service Response Code
CVV_RESPONSE	Card Verification Value Response Code
REQUEST_API_VERSION	Payment Service API Version

12.4 Service Declined Response Fields

These are the fields returned to the merchant when the transaction request has been declined by the gateway before sending the transaction to the bank.

NOTE: Transactions that have been blocked by the gateway due to invalid format or missing data will not be recorded and will not show up in merchant-facing reports on the Portal.

- REQUEST_ACTION – This will return the Service Request Action the merchant sent in the transaction request.
- TRANS_STATUS_NAME – Transaction Status.
- CUST_ID – Customer ID.
- XTL_CUST_ID – Merchant's Customer ID.
- MERCH_ACCT_ID – Inovio Account ID.
- CARD_BRAND_NAME – Bank account or credit card type (Example:" Mastercard").
- PMT_L4 – Payment account or credit card's last 4 digits.
- PMT_ID – Payment Unique Identifier
- PMT_ID_XTL – External Unique Identifier
- API_RESPONSE – Payment Service API Response Code. This is the higher level response information used for authentication and checking Service availability.
- API_ADVICE – Payment Service API Response details.
- SERVICE_RESPONSE – Service Response Code. The information in this field is for all transaction related requests (example: authorization requests and scrub-only transaction requests.)
- SERVICE_ADVICE – Service Response details.

- PROCESSOR_RESPONSE – Merchant Account Processor Response Code.
- PROCESSOR_ADVICE – Merchant Account Service Response details.
- INDUSTRY_RESPONSE – Issuing Bank's Response Code.
- INDUSTRY_ADVICE – Issuing Bank's Response details.
- REF_FIELD – Reference field (useful when getting "missing required parameter" messages).
- PROC_NAME – Processor Name (returned as Null).
- AVS_RESPONSE – Address Verification Service Response Code.
- CVV_RESPONSE – Card Verification Value Response Code.
- REQUEST_API_VERSION – Payment Service API Version.

Note

Please check the Appendix Section for a list of different response codes and descriptions.

13 Dynamic Descriptor

13.1 Dynamic Descriptor

If supported by the Merchant Account Processor, merchants may send a Dynamic Descriptor in the transaction request using the **PMT_DESCRIPTOR** parameter and/or the **PMT_DESCRIPTOR_PHONE** parameter. These parameters will replace the static statement descriptor that appears in the cardholder's bank statement.

13.1.1 Dynamic Descriptor Restrictions

In very few cases will Dynamic descriptors function properly without prior arrangements being made with your Payment Processor. Dynamic descriptors may be restricted by your payment processor. Dynamic descriptors may also have format and length restrictions. Dynamic descriptors may only take effect after a transaction has been settled. All of these issues should be reviewed with your payment processor before proceeding.

For more information on technical usage, please contact your gateway support representative about using Dynamic Descriptors.

14 Transaction Validation and Risk Assessment

14.1 Address Verification Service

The Address Verification System is a system used to verify the address provided by the customer. AVS is generally supported by most banks in the United States and some foreign countries.

By default, the Payment Service will check AVS and return the AVS response code in the AVS_RESPONSE field (as described in Service Response Fields section). Note that the response is ignored by the system unless AVS Check is enabled by the merchant.

14.1.1 Check AVS Flag (CHKAVS)

As an option, merchants may choose to ignore, disable, or even set conditions on how transactions should be approved based on the AVS response codes returned by the processor.

CHKAVS Setting	Description
T	AVS check is enabled. Note: Transaction is approved if the AVS Response Code is any of the codes described in the Positive AVS Response Codes table unless a custom **AVSMATCHSET** has been set by the merchant.
F	Do not check for AVS.
I	Check AVS but ignore the response. **(This is the default behavior.)**
C	AVS response code will be ignored based on the result of the CVV check. e.g. A transaction that returned the following response codes will be approved if CHKAVS parameter was set to "conditional": AVS RESPONSE CODE = N (No match) CVV2 RESPONSE CODE = M (MATCH

14.1.2 Positive AVS Response Codes

By default when AVS Check is enabled, the Payment Service will approve transactions that return any of the AVS Response Codes in the table below.

Important Note: Merchants may send a custom set of codes using the **AVSMATCHSET** parameter. See Section 13.1.4 below for details.

AVS Response Code	Description
A	Street address matches, but 5-digit and 9-digit postal code do not match.
B	Street address matches, but postal code not verified.
D	Street address and postal code match. Code "M" is equivalent.
E	AVS data is invalid or AVS is not allowed for this card type.
F	Card member's name does not match, but billing postal code matches.
G	Non-U.S. issuing bank does not support AVS.
H	Card member's name does not match. Street address and postal code match.

I	Address not verified.
J	Card member's name, billing address, and postal code match.
L	Card member's name and billing postal code match, but billing address does not match.
M	Street address and postal code match. Code "D" is equivalent.
O	Card member's name and billing address match, but billing postal code does not match.
P	Postal code matches, but street address not verified.
Q	Card member's name, billing address, and postal code match.
R	System unavailable.
S	Bank does not support AVS.
T	Card member's name does not match, but street address matches.
U	Address information unavailable. Returned if the U.S. bank does not support non-U.S. AVS or if the AVS in a U.S. bank is not functioning properly.
V	Card member's name, billing address, and billing postal code match.
W	Street address does not match, but 9-digit postal code matches.
X	Street address and 9-digit postal code match.
Y	Street address and 5-digit postal code match.
Z	Street address does not match, but 5-digit postal code matches.

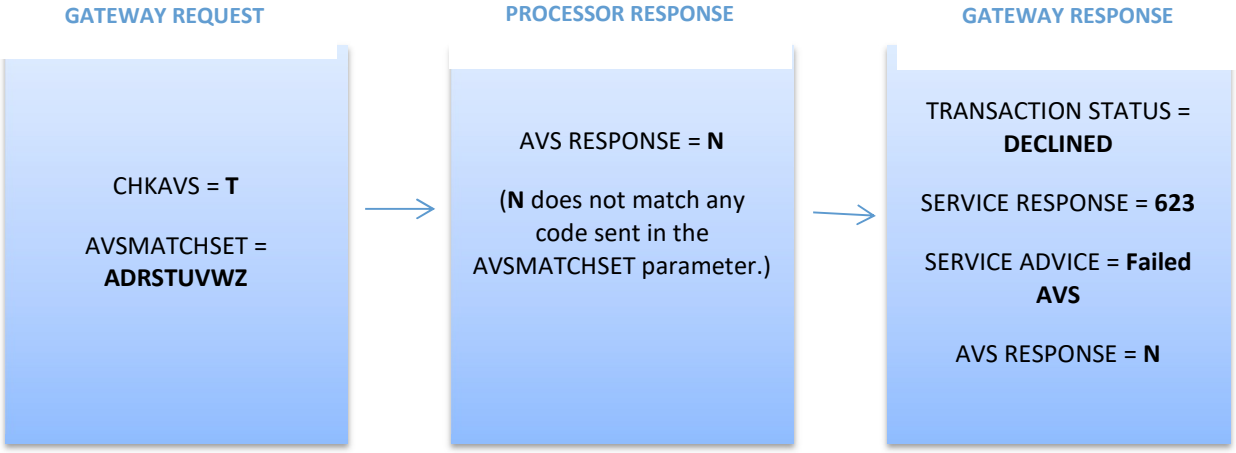
14.1.3 Negative AVS Response Codes

By default when AVS Check is enabled by the merchant, the Payment Service will decline all transactions that return any of the AVS Response Codes in the table below.

AVS Response Code	Description
C	Street address and postal code do not match.
K	Card member's name matches but billing address and billing postal code do not match.
N	Standard Response (US). Street address and postal code do not match.

14.1.4 Custom Settings (AVSMatchSet)

The "AVSMATCHSET" parameter allows merchants to send a list of AVS Response Codes. The Payment Service will return a "declined" response if the AVS response code returned by the processor **does not match** any of the codes sent in the AVSMATCHSET parameter.



14.2 Card Security Code

The Card Security Code (sometimes called CVV, CVV2, CVVC, CVC, CVC2 and V-code) is the 3 or 4-digit number located on the credit or debit card. The Payment Service requires that the merchant send these numbers in the PMT_KEY parameter by default.

Important Note: Some Merchant Account Processors may automatically decline transactions that return a negative CVV2 response. In this case, the merchant should see a "declined" transaction response message.

14.2.1 Check CVV Flag (CHKCVV)

As an option, merchants may choose to ignore, disable, or even set conditions on how transactions should be approved based on the CVV response codes returned by the processor.

CHKCVV Setting	Description
T	CVV check is enabled. Note: Transaction is approved if the CVV Response Code is any of the codes described in the Positive CVV Response Codes table unless a custom **CVVMATCHSET** has been set by the merchant.
F	Do not send CVV to the bank.
I	Check CVV but ignore the response. **(This is the default behavior.)**
C	CVV response code will be ignored based on the result of the AVS check. e.g. A transaction that returned the following response codes will be approved if CHKCVV parameter was set to "conditional": CVV2 RESPONSE CODE = N (CVV Match)

	AVS RESPONSE CODE = M (No match)

14.2.2 Positive CVV Response Codes

By default when CVV Check is enabled by the merchant, the Payment Service will approve all transactions that return any of the CVV Response Codes in the table below.

CVV Response Code	Description
M	Match
P	Not Processed
S	Not Supported
U	Service Not Available
X	No CVC/CVV/CVV2/CID Response Data Available
(empty)	No CVC/CVV/CVV2/CID Response Data Available

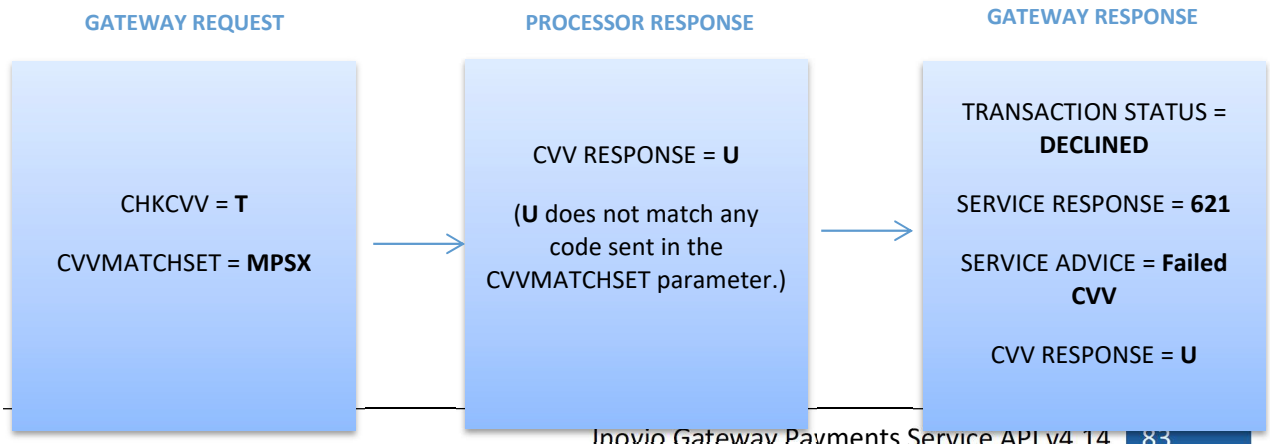
14.2.3 Negative CVV Response Codes

By default when CVV Check is enabled by the merchant, the Payment Service will decline all transactions that return any of the CVV Response Codes in the table below.

CVV Response Code	Description
N	No match

14.2.4 Custom Settings (CVVMatchSet)

The "CVVMATCHSET" parameter allows merchants to send a list of CVV Response Codes. The Payment Service will return a "declined" response if the CVV response code returned by the processor **does not match** any of the codes sent in the CVVMATCHSET parameter.



14.3 Timeout Void

The Timeout Void feature allows clients to specify a duration after which they would like to effectively "abandon" an in-flight transaction (limited to CCAUTHORIZE or CCAUTHCAP). By submitting the necessary parameters within the gateway request or by setting a global preference at the merchant account level in the portal (please contact your gateway support representative), we can enforce a time limit. If the gateway has not received an end state response from a processor within the specified time, a decline response will be sent to the client. The Timeout Void timer starts the moment the gateway receives the request.

Important Note: While the client is met with a decline response from the gateway, the transaction may still be in flight with the processor or cascade program, which could ultimately lead to an approval. If the transaction is later approved by the processor, the gateway will immediately submit a request to void the transaction at the processor level. If the authorization is declined by the processor, no further action is required.

14.3.1 Timeout Void Parameters

The below parameters allow a request to follow specific guidelines for when a client would like to force “timeout” a transaction.

Field Name	Description	Additional Notes
REQUEST_MAX_WAIT	Used to enable or disable the timeout void functionality.	1 = Enable 0 = Disable
REQUEST_MAX_WAIT_TIMER	Used to set the value for how long before the client would like to “abandon” the transaction	Value is required when “REQUEST_MAX_WAIT”=“1”. Value must be set between 30 and 600 seconds.

14.3.2 Timeout Void Portal Setting

If you would like to have a timeout setting to “abandon” every transaction submitted, please contact your gateway support representative. The product can then be enabled with the timelimit you would like to set. The portal setting can be overwritten by submitting the Timeout Void parameters previously discussed.

15 3D Secure Transactions

15.1 3D Secure

The payment gateway supports 3DS for all applicable card brands and in different scenarios. Contact your gateway support respresetnative for more information.

When integrating and testing 3-D Secure (3DS), it's crucial to use specific test card numbers and scenarios to ensure the authentication flows are working correctly. For a comprehensive list of

test card numbers and specific test case details, https://developer.cardinaltrusted.com/reference/emv-3ds-test-copy.

15.1.1 Low Value Strong Customer Authentication (SCA) Exemption

Where Strong Customer Authentication (SCA) is mandated but transactions qualify for SCA Exemptions for reasons like low value transactions, the Gateway has no way to know when such exemptions have been met. Where the exemption is supported by the Processor, the gateway shall make available the REQ_LOW_VALUE_SCAEXEMPTION parameter for merchants to use when they want a transaction to bypass SCA.

15.1.2 External 3DS Provider Scenario

In this scenario, the merchant uses an external 3DS provider, and the 3DS authentication is completed before authorization requests are sent to the gateway.

15.1.2.1 3DS Authentication Data Gateway Parameters.

The payment gateway has defined parameters for 3DS on authorization request and these parameters are named in accordance with the 3DS standards which have been established in the industry. These are described in the table below. These parameters should be sent (with data provided by the external 3DS provider) along with the authorization request.

Field Name	Description
P3DS_CAVV	Cardholder Authentication Verification Value
P3DS_ECI	Electronic Commerce Indicator
P3DS_XID	Transaction ID
P3DS_VERSION	Reports on 3DS Version used to process Transaction (Required for Mastercard Identity Check transactions in Authorization on 3DS 2)
P3DS_TRANSID	Unique transaction identifier assigned by the Directory Server (DS)-(Required for Mastercard Identity Check transactions in Authorization IF P3DS_VERSION is 3DS 2)
P3DS_SCREEN_HEIGHT	Total height of the cardholder's screen in pixels
P3DS_SCREEN_WIDTH	Total width of the cardholder's screen in pixels
P3DS_JAVA_ENABLED	A Boolean value (TRUE/FALSE) that represents the ability of the cardholder browser to execute Java
P3DS_JAVASCRIPT_ENABLED	A Boolean value (TRUE/FALSE) that represents represents the ability of the cardholder browser to execute JavaScript
P3DS_BROWSER_HEADER	The exact content of the HTTP accept headers sent from the cardholder's browser. Example: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
P3DS_BROWSER_LANGUAGE	Value represents the browser language as defined in IETF BCP47
P3DS_BROWSER_COLOR_DEPTH	Value represents the bit depth of the color palette for displaying images, in bits per pixel Possible Values: 1, 4, 8, 15, 16, 24, 32, 48

P3DS_BROWSER_TIME_ZONE	Time difference between UTC time and the cardholder browser local time, in minutes Note: Regardless of direction value should be positive.
P3DS_CHALLENGE_WINDOW	An override field that a merchant can pass in to set the challenge window size to display to the end cardholder. The ACS will reply with content that is formatted appropriately to this window size to allow for the best user experience. The sizes are width x height in pixels of the window displayed in the cardholder browser window. Possible values: 01 - 250x400, 02 - 390x400, 03 - 500x600, 04 - 600x400, 05 - Full page
USER_AGENT_XTL	Software agent responsible for retrieving and facilitating end-user interaction with Web content
XTL_IP	Cardholder's IP Address

15.1.3 Payment Gateway as 3DS Provider Using Redirect Scenario

The following steps and examples provide a guide on how to use 3DS 2.0 with Inovio's payment gateway through a Merchant's own payment page:

15.1.3.1 Make a request to Inovio's API to retrieve Device Data Collection (DDC) references and a JWT

15.1.3.1.1 Inovio 3DS API request parameters and example - See step A in the process flow below

Endpoint: https://api.inoviopay.com/payment/3dsrequest.cfm

Parameters:

Field Name	Description
REQ_USERNAME	API credential username
REQ_PASSWORD	API credential password
MERCH_ACCT_ID	Numeric merchant account identifier
REQUEST_API_VERSION	Version of the API
BILL_ADDR_COUNTRY*	2-letter ISO 3166-1 Alpha-2 code
REQUEST_CURRENCY*	3-letter ISO-4217 code
SITE_ID*	Merhcnat's site identifier
PMT_BIN	First 6 digits of the card number
CUST_ID	Customer ID Required when using a tokenized card. Cannot be sent with PMT_BIN
PMT_ID	Payment ID Required when using a tokenized card. Cannot be sent with PMT_BIN

P3DS_SCREEN_HEIGHT	Total height of the cardholder's screen in pixels
P3DS_SCREEN_WIDTH	Total width of the cardholder's screen in pixels
P3DS_JAVA_ENABLED	A Boolean value (TRUE/FALSE) that represents the ability of the cardholder browser to execute Java
P3DS_JAVASCRIPT_ENABLED	A Boolean value (TRUE/FALSE) that represents represents the ability of the cardholder browser to execute JavaScript
P3DS_BROWSER_HEADER	The exact content of the HTTP accept headers sent from the cardholder's browser. Example: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
P3DS_BROWSER_LANGUAGE	Value represents the browser language as defined in IETF BCP47
P3DS_BROWSER_COLOR_DEPTH	Value represents the bit depth of the color palette for displaying images, in bits per pixel Possible Values: 1, 4, 8, 15, 16, 24, 32, 48
P3DS_BROWSER_TIME_ZONE	Time difference between UTC time and the cardholder browser local time, in minutes Note: Regardless of direction value should be positive.
P3DS_CHALLENGE_WINDOW	An override field that a merchant can pass in to set the challenge window size to display to the end cardholder. The ACS will reply with content that is formatted appropriately to this window size to allow for the best user experience. The sizes are width x height in pixels of the window displayed in the cardholder browser window. Possible values: 01 - 250x400, 02 - 390x400, 03 - 500x600, 04 - 600x400, 05 - Full page
USER_AGENT_XTL	Software agent responsible for retrieving and facilitating end-user interaction with Web content
XTL_IP	Cardholder's IP Address

* = REQUIRED IF **MERCH_ACCT_ID** IS NOT PROVIDED

```
/payment/3dsrequest.cfm?req_username=api%40base.prod&req_password=Pass&site_id=70&merch_acct_id=10&pmt_numb=400000
```

15.1.3.1.2 Inovio 3DS API response example- See step B in the process flow below

```
{
 "REQ_ID": "28178",
 "JWT": "eyJhbGciOiJIUzI1NiJ9.eyJpY /Ssy7eJmG2kLg",
 "PMT_BIN": "400000",
 "MERCH_ACCT_ID": "1288",
 "DDC_URL": "https://cas.client.cardinaltrusted.com/centinelapi/V2/Cruise/Collect",
 "DDC_REFERENCEID": "2968-148"
}
```

Note:

- JWT, DDC_REFERENCEID, AND PMT_BIN WILL BE USED IN THE NEXT STEP.

15.1.3.2 DDC Starts - See step C in the process flow below

DDC takes place in an iframe and requires the JWT and PMT_BIN number of the customer's card. This form must be submitted **before** the checkout form is submitted.

```
<form name="ddcForm" id="ddcForm" method="POST">
  <input type="hidden" name="JWT" id="JWT"/>
  <input type="hidden" name="Bin" id="Bin"/>
  <input type="hidden" name="DDC_REFERENCEID" id="DDC_REFERENCEID"/>
</form>
```

An eventListener should be added so that the DDC can finish before proceeding with submission of the checkout form. An example of this is as follows:

```
const trustedOrigins = [
  "https://centinelapi.cardinalcommerce.com",
  "https://centinelapistag.cardinalcommerce.com/",
  "https://client.cardinaltrusted.com",
  "https://cas.client.cardinaltrusted.com/",
  "https://another-trusted-domain.com",
];
request.onload = function (req) {
   var json = JSON.parse(req.currentTarget.response || req.target.responseText);
   window.addEventListener("message", function(event) {
      if (trustedOrigins.includes(event.origin)) {
          //Here you call your server side to make the call to 3dsrequest.cfm);
       } else {
          // The origin is not trusted, do not process the message
          console.error("Message blocked from an untrusted origin:", event.origin);
       }
   }, false);
   //populate DDC form
};
request.send();
```

NOTE: Grant time for the DDC to complete for potentially increased approval rates.

15.1.3.3 DDC Complete

After the DDC finishes and the checkout form is submitted, the merchant should send a transaction request to Inovio's gateway API.

15.1.3.3.1 Inovio Payment Gateway API Request - *See step D in the process flow below*

Endpoint: https://api.inoviopay.com/payment/pmt_service.cfm

Please refer to the Inovio Gateway API Documentation for full usage and parameter details Request parameters that are specific to 3DS and specific values to use are:

Field Name	Description
REQUEST_ENROLLMENT	TELLS THE GATEWAY TO CHECK THE CARD'S 3DS ENROLLMENT 0 (off) 1(on)
DDC_REFERENCEID	The Device Data Collection ID to be used to confirm the customers' details collected and shared with the 3DS provider.

15.1.3.3.1.1 Request Example

```
/payment/pmt_service.cfm?req_username=api%40base.dev&req_password=Pass&site_id=12&merch_acct_id=148&li_prod_id_1=12&li_value_1=10.00&cust_fname=Mister&cust_lname=Tester&cust_email=mtester%403dstest.dev&bill_addr=123+Main+Street&bill_addr_city=Los+Angeles&bill_addr_state=CA&bill_addr_zip=90001&pmt_numb=400000000000####&pmt_key=123&pmt_expiry=09%2F2028&request_action=CCAUTHCAP&request_response_format=JSON&request_api_version=4.14&bill_addr_country=US&request_currency=USD&domainName=acme.com&p3ds_return_url=https%3A%2F%2FtInovio.com%2F3DSReturn&request_enrollment=1&ddc_referenceid=28194220-144288
```

15.1.3.3.1.2 Response Parameters

Field Name	Description
PROC_REDIRECT_URL	3D Secure Verification URL where the customer should be redirected to for authentication.
P3DS_TRANSID	3D Secure Vendor's Transaction ID
P3DS_PAREQ	Payment Authentication Request

15.1.3.3.1.3 Response Example See step F in the process flow below


```
{
  "REQUEST_ACTION":"CCAUTHCAP",
  "REQ_ID":"4888098",
  "TRANS_STATUS_NAME":"PENDING",
...
... [Other response parameters as described in Inovio Gateway API documentation]
...
"PROC_REDIRECT_URL":"https://3ds.example.com/merchantacsfrontend/pareq.jsp?vaa=b&gold=AAAA",
  "P3DS_RESPONSE":"Full Authentication",
  "P3DS_VENDOR":"Cardinal",
  "P3DS_ECI":"07",
  "P3DS_TRANSID":"EnK4GI8WOhkBRVzQDUIo",

"P3DS_PAREQ":"eNpVUV1vgjAUfe+vIGbPtAVUYq5N/JpjH+pQNNkbgWaQadFSjNuvX4sw3ds9997TnnMubDLJ+XTNk0pyBm+8LONPbuXpsDMTL9488HfL7Gscbn/ep1FAOgxWo5CfGJy5LPNCMGoT2wHcQqSfkEkWC8UgTk7jYMGo43rdXt8H3DQQHLgMpneD8i3u4hf74h57yw2oesDS8V4HqGICkqoeQ38zoCuAUIKrlnmVLHcoBxLgzfTooDYNNHgG+iVpWpSu31kqcs3HVfF7P0efmYruK5mkT7j14Uhdv1ZjQEbDYQpLHizCHUJz1CLeoPPDIgfcB1H0F8MCLYg2sTrahBCI7mn9EVuWZy39A2Kim5SFofLULAL8dCcL2jQ/2rtYOb7smTiTZRJkHX9SilJtka1/RcB+IQQmt+XqeDDQc3l8PNkXX17/i/rCWomA==",
}
```


15.1.3.3.2 Inovio Payment Gateway API Response

There are two possible outcomes in the response:

- Outcome 1: 3DS authentication will be required
- Outcome 2: In some cases, the issuer will allow a "frictionless" authentication, or the issue will not support any authentication; in either of these cases, the transaction will be fully completed with no further action needed.

15.1.4 Determining which outcome happened

The table shows how to determine which of the three outcomes happened, according to parameters returned by the API: TRANS_STATUS_NAME and P3DS_PROCVERSION

	TRANS_STATUS_NAME
Outcome 1	PENDING

Outcome 2 (no action)	APPROVED or DECLINED

15.1.4.1 Outcome 1: 3DS authentication - See step F in process flow below


```
{
"REQUEST_ACTION": "CCAUTHCAP",
"REQ_ID": "5551043",
...
... [Other response parameters as described in Inovio Gateway API documentation]
...
"TRANS_STATUS_NAME": "PENDING",
"PROC_REDIRECT_URL":https://centinelapistag.cardinalcommerce.com/V2/Cruise/StepUp",
"P3DS_RESPONSE": "Full Authentication",
"P3DS_JWT":"eyJhbGciOiJIUzI1NiJ9.eyJpYXQiOjE2MDgwNjQ4NzEsIk9yZ1VuaXRJZCI6IjVhZjRhNTRiMGU
0MjNkMjY4ODA4MmRjZCIsIk9iamVjdGlmeVBheWxvYWQiOiB0cnVlLCJpc3MiOiI1ZDI3OThiMjZmZTNkM
TIxNWNjNmQ4OGMiLCJqdGkiOiI1NTUxMDQzIiwiUmV0dXJuVXJsIjoiaHR0cDovL2xvY2FsaG9zdDo4MDg
wL3RocmVlZHN0ZXN0XzFfMF93YXIvdGhyZWVEU1JldHVybiIsIlJlZmVyZW5jZUlkIjoiNTU1MTA0MS0yMD
AxMyIsIlBheWxvYWQiOnsiQUNTVXJsIjoiaHR0cHM6Ly8wbWVyY2hhbnRhY3NzdGFnLmNhcmRpbmFsY2
9tbWVyY2UuY29tL01lcmNoYW50QUNTV2ViL2NyZXEuanNwIiwKICAgICAgICAgICAgICAgICAgICAgICAgI
CAgICAgICAgICAgICAgICAgICAgICAgICAgICAiUGF5bG9hZCI6ImV5SnRaWE56WVdkbFZIbHdaU0k2SWt
OU1pYRWlMQ0p0WlhOellXZGxWbVZ5YzJsdmJpSTZJakl1TVM0d0lpd2lkR2h5WldWRVUxTmxjblpsY2xSeV
lXNXpTVVFpT2lJNU16RmxabVZpWmkweFlqWmtMVFF3Wm1NdFlXSTJaUzFtTTJRME9XSTRNelUwWmp
RaUxDSmhZM05VY21GdWMwbEVJam9pTW1JeFpURTRaakFoWkRrMU1TMDBOMk01TFdGa01HTXRZamd
6WTJOaU9EQTFZVFl6SWl3aVkyaGhiR3hsYm1kbFYybHVaRzkzVTJsNlpTSTZJakF5SW4wIiwKICAgICAgIC
AgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAiVHJhbnNhY3Rpb25JZCI6Ildn
TlZOMjRmMmZ3aDFCOERnVoUwIn19.AvfAiDc5XHU1V8or2XdyCaMOD8SDG9ZVQvLo+s0tfGA",
"P3DS_PROCVERSION": "2.2.0",
"P3DS_PROCTRANSID": "WgNVN24f2fwh1B8DgWE0"
}
```


15.1.4.1.1 3D Secure Full Authentication Purchase

Endpoint: https://api.inoviopay.com/payment/pmt_service.cfm

15.1.4.1.1.1 Request Parameters

Field Name	Description
P3DS_TRANSID	Transaction ID returned in first response, such as in the above example: P3DS_TRANSID:EnK4GI8WOhkBRVzQDUI0
REQUEST_PARES	The value returned from the enrollment check, which is passed to the designated **TERMURL** described in Step 4 above

15.1.4.1.1.2 3D Secure Purchase Example


```
/payment/pmt_service.cfm
?cust_fname=James&cust_lname=Sample&pmt_expiry=012020&li_count_1=1&li_prod_id_1=32805
&li_value_1=3.00&xtl_udf01=udfex&bill_addr_country=US&merch_acct_id=25109&request_cur
rency=USD&P3DS_TRANSID=EnK4GI8WOhkBRVzQDUI0&REQUEST_PARES=eNqdmFuTqsiygN%2BN8D%2BsmP3
ozACKtwm7IwoobgrKXXzjJncQQUB%2B%2FS519eqetVfsPef0SxdpVlZWZeZXCRs9ugYBowXe7Rq8b6Sgrp0w
%2BBb7b7%2FBYktywsraRymlmoPCGAL%2B2%2FvmANSgfio8R8suECrdiwI6uJ26lXOYDEWWFTQp1m9Iuw2ud
Vw&req_username=api%example.com&req_password=Password123&site_id=12345&request_respon
se_format=JSON&request_action=CCAUTHCAP&request_api_version=4.14&pmt_numb=40000000000
00002&PROC_SUCCESS_URL=http://www.example.com&PROC_ERROR_URL=http://www.example.com
```


15.1.4.1.1.3 3D Secure Response Parameters

Merchants using 3D Secure Verification will receive these additional parameters in the gateway response.

Field Name	Description

SERVICE_REPONSE	This is the standard gateway service response, but may contain a 3DS-specific code in certain decline cases
SERVICE_ADVICE	This is the standard gateway service response, but may contain 3DS-specific information in certain decline cases
P3DS_RESPONSE	3D Secure service response
P3DS_VENDOR	3D Secure downstream provider *(informational only)*

- After the payment gateway request is completed, the actual 3DS authentication process can proceed on the checkout page.

ALL OF THE FOLLOWING STEPS FOLLOW FROM OUTCOME 1 IN THE PRECEDING SECTION

- A form will need to be submitted to the PROC_REDIRECT_URL with the parameters shown in the below image.

Note:

SEE THE DATA IN G IN THE PROCESS FLOW BELOW

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
  <script type="text/javascript">
     window.onload = function () {
        document.forms['stepUpForm'].submit();
     }
  </script>
</head>
<body>
  <form id="stepUpForm" method="post" action="${url}">
     <input type="hidden" name="JWT" value="${jwt}" />
     <input type="hidden" name="MD" value="${md}" />
  </form>
</body>
</html>
```

FORM POST ACTION

//THIS SHOULD BE THE VALUE OF PROC_REDIRECT_URL FROM THE INOVIO API RESPONSE DESCRIBED ABOVE

HIDDEN INPUT FIELD JWT

//THIS SHOULD BE THE VALUE OF P3DS_JWT FROM THE INOVIO API RESPONSE DESCRIBED ABOVE

HIDDEN INPUT FIELD MD

//THIS IS AN OPTIONAL PARAMETER WHICH CAN BE USED BY THE MERCHANT TO STORE ITS OWN REFERENCE DATA/IDENTIFIER

- After the above form is submitted to the PROC_REDIRECT_URL, the user will authenticate with the ACS, and then be redirected back to the merchant's website via the P3DS_RETURN_URL.

See step H in the process flow below

- The redirect back from PROC_REDIRECT_URL to P3DS_RETURN_URL will be a POST, and will contain the POST parameters TRANSACTIONID, RESPONSE, and MD. At this point, the

merchant should use the data in those parameters to make a final request to Inovio Gateway API to perform a transaction.

See step I in the process flow below

Please refer to the Inovio's Gateway API Documentation for full usage and parameter details

Request parameters that are specific to 3DS and specific values to use are:

P3DS_PROCTRANSID

//THIS SHOULD BE THE VALUE OF TRANSACTIONID FROM THE ACS

REQUEST_PARES

//THIS SHOULD BE THE VALUE OF RESPONSE FROM THE ACS

Note:

THE VALUE OF THE "RESPONSE" PARAMETER FROM THE REDIRECT BACK TO THE RETURN URL MAY BE EMPTY. THIS IS NOT AN ERROR. THAT EMPTY VARIABLE SHOULD BE PASSED AS THE EMPTY VALUE FOR REQUEST_PARES ON THE SUBSEQUENT GATEWAY REQUEST.

THE RESPONSE FROM THE GATEWAY API WILL CONTAIN THE RESULT OF THE TRANSACTION. IF SUCCESSFUL, THE TRANSACTION WILL HAVE THE 3DS AUTHENTICATION ATTACHED TO IT AT THE PROCESSOR.

See the data in J in the process flow below

15.1.4.2 Outcome 2: No further action needed – See the question before step F in the process flow below

```
{
 "REQUEST_ACTION":"CCAUTHCAP",
 "REQ_ID":"28194338",
...
... [Other response parameters as described in Inovio Gateway API documentation]
...
 "TRANS_STATUS_NAME":"Approved",
 "P3DS_RESPONSE":"Frictionless Authentication",
 "P3DS_TRANSID":"e21-67801d7ab7e1"
}
```

15.1.5 Special cases

15.1.5.1 Using Inovio API 3DS 2.0 Guide with Ephemoral Tokenized Cards

THIS IS USED FOR ANY CIT (CUSTOMER INITIATED TRANSACTION) USING AN EPHEMORAL TOKEN OR "ONE-CLICK" WITH 3DS 2.0

To use ephermoral tokenized cards, adhere to the following practices in the documented flow:

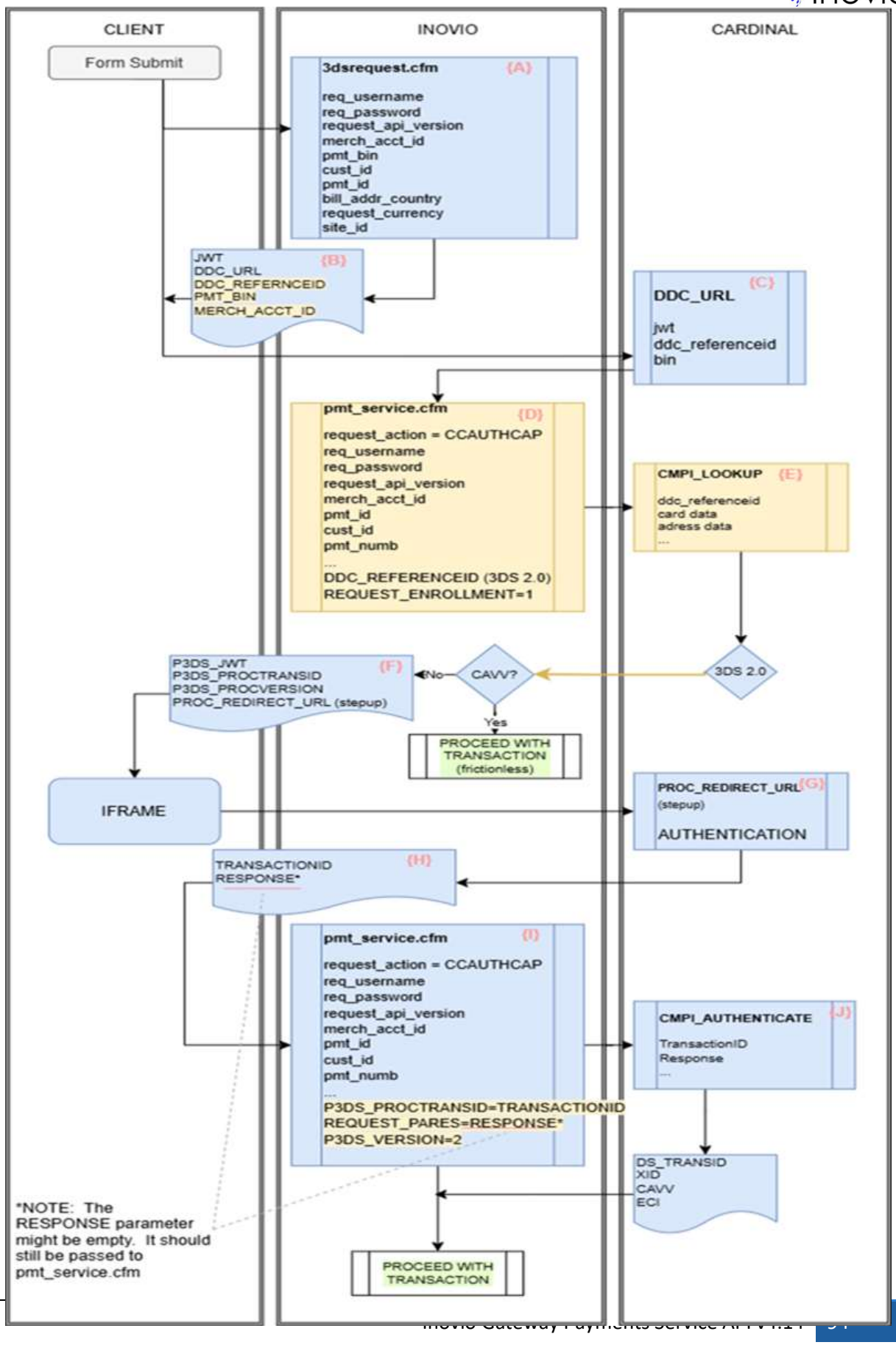
- Make sure to send CUST_ID and PMT_ID values when making request to **https://api.inoviopay.com/payment/3dsrequest.cfm**
  This will allow Inovio's system to look up the card value and provide the PMT_BIN value which will be necessary in later steps
- Make sure to capture the PMT_BIN value from the response from your request to **https://api.inoviopay.com/payment/3dsrequest.cfm**
- Make sure to send the same CUST_ID and PMT_BIN values to the gateway API **https://api.inoviopay.com/payment/3dsrequest.cfm** in all subsequent steps. See the Inovio's Payment Gateway API documentation for more details.

15.1.5.2 Using Inovio API 3DS 2.0 Guide with Inovio's Merchant Account Distribution

Inovio's merchant account distribution enbles merchant to intelligently use all of their merchant accounts and route transactions to the appropriate merchant account, based on card issuer, locale, currency, card type, and other available data.

To use Merchant Account Distribution, adhere to the following practices in the documented flow:

- Note that the value for MERCH_ACCT_ID will not be known at the start of this process, but will be determined in the first step, and then must be captured and used in later steps.
- Be sure to send BILL_ADDR_COUNTRY, REQUEST_CURRENCY, and SITE_ID values when making request to **https://api.inoviopay.com/payment/3dsrequest.cfm**
  This will allow Inovio's system to look up the card value and provide the MERCH_ACCT_ID value which will be necessary in later steps
- Make sure to capture the MERCH_ACCT_ID value from the response from **https://api.inoviopay.com/payment/3dsrequest.cfm**
- Make sure to send the MERCH_ACCT_ID value to the gateway API **https://api.inoviopay.com/payment/3dsrequest.cfm** in all subsequent steps. See the Inovio Payment Gateway API documentation for more details.


CLIENT
INOVIO
CARDINAL
Form Submit
3dsrequest.cfm {A}
req_username
req_password
request_api_version
merch_acct_id
pmt_bin
cust_id
pmt_id
bill_addr_country
request_currency
site_id
JWT {B}
DDC_URL
DDC_REFERNCEID
PMT_BIN
MERCH_ACCT_ID
DDC_URL {C}
jwt
ddc_referenceid
bin
pmt_service.cfm {D}
request_action = CCAUTHCAP
req_username
req_password
request_api_version
merch_acct_id
pmt_id
cust_id
pmt_numb
...
DDC_REFERENCEID (3DS 2.0)
REQUEST_ENROLLMENT=1
CMPI_LOOKUP {E}
ddc_referenceid
card data
adress data
...
3DS 2.0
CAVV?
No
Yes
P3DS_JWT {F}
P3DS_PROCTRANSID
P3DS_PROCVERSION
PROC_REDIRECT_URL (stepup)
PROCEED WITH TRANSACTION (frictionless)
IFRAME
PROC_REDIRECT_URL {G}
(stepup)
AUTHENTICATION
TRANSACTIONID {H}
RESPONSE*
pmt_service.cfm {I}
request_action = CCAUTHCAP
req_username
req_password
request_api_version
merch_acct_id
pmt_id
cust_id
pmt_numb
...
P3DS_PROCTRANSID=TRANSACTIONID
REQUEST_PARES=RESPONSE*
P3DS_VERSION=2
CMPI_AUTHENTICATE {J}
TransactionID
Response
...
DS_TRANSID
XID
CAVV
ECI
*NOTE: The RESPONSE parameter might be empty. It should still be passed to pmt_service.cfm
PROCEED WITH TRANSACTION

15.1.6 3DS Browser & Device Data

15.1.6.1 Overview

The Inovio payment gateway now supports passing browser and device data fields directly in the Step 3 enrollment request (CMPI Lookup) to pmt_service.cfm. These fields are forwarded to Cardinal Commerce for 3DS2 risk evaluation, which can improve approval rates by providing Cardinal with richer device context.

This enhancement provides two key benefits:

- Improved approval rates — Cardinal uses browser metadata for more accurate risk scoring, increasing the likelihood of frictionless authentication.
- DDC resilience — Even if the Device Data Collection iframe (Step 2) fails or is blocked by the cardholder's browser, Cardinal still receives essential device metadata for risk evaluation.

Note: This feature is optional. Your existing 3DS integration will continue to work without modification. Adding browser fields enhances the data available to Cardinal but is not required for basic 3DS functionality.

Where to send these fields: Browser data fields are sent *only* in Step 3 (the enrollment/CMPI Lookup request to PMT_SERVICE.CFM with REQUEST_ENROLLMENT=1). They are *not* required in Step 1 (3DSREQUEST.CFM) or Step 5 (final authentication).

15.1.6.2 Browser & Device Data Fields Reference

The gateway accepts the following browser/device fields in the Step 3 enrollment request. The gateway automatically adds two hardcoded values to the Cardinal lookup: DeviceChannel: "Browser" and BrowserJavascriptEnabled: true. You do not need to send these.

15.1.6.2.1 Required Fields (Minimum for 3DS Activation)

All three of the following fields must be present in the request for Cardinal to perform 3DS authentication. If any one is missing, the transaction skips 3DS entirely and processes as a normal non-3DS authorization.

Important: Cardinal enforces the presence of these three fields, not their value correctness. If any required field is missing, Cardinal returns an error message identifying the missing field and the transaction falls back to non-3DS authorization.

Field Name	Description
P3DS_BROWSER_LANGUAGE	The browser language as defined in IETF BCP47. Example: en-US
USER_AGENT_XTL	The full User-Agent string of the cardholder's browser. Example: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
P3DS_BROWSER_HEADER	The HTTP Accept header sent from the cardholder's browser. Example: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

15.1.6.2.2 Optional Fields

The following fields provide additional device context to Cardinal for risk scoring. They may be included in any combination and do not affect whether 3DS authentication occurs. Omitting or sending malformed values for these fields will not trigger errors.

Field Name	Type	Description
P3DS_JAVA_ENABLED	Boolean	Whether the cardholder's browser can execute Java. Values: true or false
P3DS_BROWSER_COLOR_DEPTH	Integer	Bit depth of the color palette for displaying images, in bits per pixel. Possible values: 1, 4, 8, 15, 16, 24, 32, 48
P3DS_SCREEN_HEIGHT	Integer	Total height of the cardholder's screen in pixels. Example: 1080
P3DS_SCREEN_WIDTH	Integer	Total width of the cardholder's screen in pixels. Example: 1920
P3DS_BROWSER_TIME_ZONE	Integer	Time difference between UTC and the cardholder's browser local time, in minutes. Example: 480 (US Pacific)
P3DS_CHALLENGE_WINDOW	String	Override field to set the challenge window size displayed to the cardholder. The ACS will reply with content formatted to this size. Values: 01 – 250x400, 02 – 390x400, 03 – 500x600, 04 – 600x400, 05 – Full page
P3DS_IP_ADDRESS	String	The cardholder's IP address. If omitted, the gateway uses the client IP from the request. Example: 203.0.113.42

15.1.6.3 Integration Instructions

Browser data fields are added to the existing 3DS 2.0 redirect flow documented in Section 15.1.3 of the Inovio Gateway Payment Service API specification. The only change is in Step D (the enrollment request to pmt_service.cfm).

15.1.6.3.1 Standard Flow (with DDC)

Your existing 5-step flow remains the same, with browser fields added to Step 3:

Step	Action	Change
1 (A/B)	Request DDC references from 3dsrequest.cfm	No change
2 (C)	Send DDC to Cardinal via iframe	No change
3 (D)	Enrollment request to pmt_service.cfm	Add browser data fields to this request alongside your existing parameters and REQUEST_ENROLLMENT=1

4 (G/H)	Challenge step-up (if PENDING)	No change
5 (I/J)	Final authentication to pmt_service.cfm	No change — do not send browser fields in this step

15.1.6.3.2 Simplified Flow (Skip DDC)

Because the browser data fields provide Cardinal with device context independently of the DDC iframe, you may optionally skip Step 2 (Device Data Collection) entirely. This is useful when:

- The cardholder's browser blocks third-party iframes or cookies
- You want to reduce latency by eliminating the DDC round-trip
- Your architecture makes it difficult to embed the DDC iframe

In this simplified flow:

Step	Action	Notes
1	Request DDC references from 3dsrequest.cfm	Still required — you need the DDC_REFERENCEID
2	Send DDC to Cardinal via iframe	SKIP — Do not POST to DDC_URL
3	Enrollment request to pmt_service.cfm	Include DDC_REFERENCEID from Step 1 and browser data fields
4	Challenge step-up (if PENDING)	Same as standard flow
5	Final authentication	Same as standard flow

Note: Even when skipping DDC, you must still send the DDC_REFERENCEID obtained from Step 1 in your Step 3 request. The DDC_REFERENCEID is generated by the gateway, not by the Cardinal DDC process.

15.1.6.4 Request Examples

15.1.6.4.1 Step 3 Enrollment Request with Browser Fields

Endpoint: https://api.inoviopay.com/payment/pmt_service.cfm

```
/payment/pmt_service.cfm
?req_username={your_api_username}
&req_password={your_api_password}
&site_id={your_site_id}
&merch_acct_id={your_merch_acct_id}
&li_prod_id_1={product_id}
&li_value_1=10.00
&cust_fname=John
&cust_lname=Smith
&cust_email=jsmith@example.com
&bill_addr=123+Main+Street
&bill_addr_city=Los+Angeles
&bill_addr_state=CA
&bill_addr_zip=90001
```

```
&bill_addr_country=US
&pmt_numb=4000000000002503
&pmt_key=123
&pmt_expiry=12/2028
&request_action=CCAUTHCAP
&request_response_format=JSON
&request_api_version=4.14
&request_currency=USD
&request_enrollment=1
&ddc_referenceid={from_step_1}
&p3ds_return_url=https://yoursite.com/3ds-return
&p3ds_browser_language=en-US
&user_agent_xtl=Mozilla/5.0+(Windows+NT+10.0;+Win64;+x64)+AppleWebKit/537.36
&p3ds_browser_header=text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
&p3ds_java_enabled=false
&p3ds_browser_color_depth=24
&p3ds_screen_height=1080
&p3ds_screen_width=1920
&p3ds_browser_time_zone=480
&p3ds_challenge_window=05
```

15.1.6.4.2 Step 3 Response (Challenge Required)

When Cardinal requires cardholder authentication:

```
{
 "REQUEST_ACTION": "CCAUTHCAP",
 "REQ_ID": "72901514548",
 "TRANS_STATUS_NAME": "PENDING",
 "MERCH_ACCT_ID": "{your_merch_acct_id}",
 "CURR_CODE_ALPHA": "USD",
 "PROC_REDIRECT_URL": "https://cas.client.cardinaltrusted.com/centinelapi/V2/Cruise/StepUp",
 "P3DS_VENDOR": "Cardinal",
 "P3DS_RESPONSE": "Full Authentication",
 "P3DS_JWT": "{jwt_token}",
 "P3DS_PROCVERSION": "2.2.0",
 "P3DS_PROCTRANSID": "{transaction_id}",
 "REQUEST_API_VERSION": "4.14"
}
```

When TRANS_STATUS_NAME is PENDING, proceed to Step 4 (challenge step-up) as documented in the main API specification.

15.1.6.4.3 Step 3 Response (Frictionless)

When Cardinal approves without requiring cardholder interaction:

```
{
 "REQUEST_ACTION": "CCAUTHCAP",
 "REQ_ID": "28194338",
 "TRANS_STATUS_NAME": "Approved",
 "P3DS_RESPONSE": "Frictionless Authentication",
 "P3DS_TRANSID": "e21-67801d7ab7e1"
}
```

When TRANS_STATUS_NAME is APPROVED, the transaction is complete. No further 3DS steps are needed.

15.1.6.5 Collecting Browser Data (JavaScript Reference)

The following JavaScript example shows how to collect browser data fields from the cardholder's browser. Run this on the client side (in the browser) and include the values in your server-side Step 3 request.

```
// Collect browser data on the client side
var browserData = {
 p3ds_browser_language: navigator.language || "en-US",
 user_agent_xtl:       navigator.userAgent,
 p3ds_browser_header:
   "text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8",
 p3ds_java_enabled:     navigator.javaEnabled ? String(navigator.javaEnabled()) : "false",
 p3ds_browser_color_depth: String(screen.colorDepth),
 p3ds_screen_height:    String(screen.height),
 p3ds_screen_width:     String(screen.width),
 p3ds_browser_time_zone: String(new Date().getTimezoneOffset()),
 p3ds_challenge_window:  "05"
};
```

Note: P3DS_BROWSER_HEADER: The browser's actual Accept header is not directly accessible via JavaScript. Use a representative value as shown above, or capture the Accept header server-side from the cardholder's initial page request.

Note: P3DS_BROWSER_TIME_ZONE: getTimezoneOffset() returns the offset in minutes. For example, US Pacific (UTC-8) returns 480. Send the value as-is.

15.1.6.6 Validation & Error Handling

15.1.6.6.1 Missing Required Fields

If any of the three required fields (P3DS_BROWSER_LANGUAGE, USER_AGENT_XTL, P3DS_BROWSER_HEADER) is missing from the request:

- The transaction skips 3DS authentication entirely
- The payment processes as a normal non-3DS authorization
- The transaction is APPROVED (no 3DS challenge or friction)
- Cardinal returns an error in P3DS_RESPONSE identifying the missing field

Example error response when USER_AGENT_XTL is missing:

```
"P3DS_RESPONSE": "A message element required as defined in Table A.1 is missing from the message.
browserUserAgent"
```

The final word in the error message changes depending on which field is missing:

Missing Field	Cardinal Error Message Suffix
P3DS_BROWSER_LANGUAGE	browserLanguage
USER_AGENT_XTL	browserUserAgent

P3DS_BROWSER_HEADER	browserAcceptHeader

15.1.6.6.2 Value Validation

Cardinal enforces presence, not value correctness:

- No value validation is performed on any of the browser fields
- Any string, integer, or boolean value is accepted
- Optional fields may be omitted, malformed, or empty without triggering errors

15.1.6.7 Testing

15.1.6.7.1 Test Cards

Use the following test card numbers with the Inovio test environment:

Card Number	Scenario	Expected Behavior
4000000000002503	Challenge	TRANS_STATUS_NAME: PENDING Triggers Step 4 challenge step-up
4000000000002000	Frictionless	TRANS_STATUS_NAME: Approved Skips Step 4, completes immediately

Test card details: Expiry: 12/2028 (MM/YYYY), CVV: 123

15.1.6.7.2 Verification Checklist

#	Test Case	Expected Result
1	Send all 3 required browser fields + optional fields in Step 3	TRANS_STATUS_NAME: PENDING with challenge flow (card 2503)
2	Send all 3 required browser fields, skip DDC (Step 2)	TRANS_STATUS_NAME: PENDING — works without DDC
3	Omit one required field (e.g. USER_AGENT_XTL)	TRANS_STATUS_NAME: APPROVED with non-3DS auth; P3DS_RESPONSE contains missing field error
4	Send no browser fields (existing integration, no changes)	Existing behavior unchanged — 3DS via DDC only

16 Currency

16.1 Multi-currency Support

The payment gateway supports **all** foreign currencies; however, availability to settle transactions in different currencies depends on the merchant account provider. Send an inquiry to your gateway support representative before implementing non-US Dollar transactions in your system.

16.2 Request_currency Parameter

To set the currency requested in your transaction request, send the 3-letter ISO-4217 (http://en.wikipedia.org/wiki/ISO_4217) currency code in the **request_currency** parameter.

16.3 Currency Response Fields

The table below explains currency related gateway response fields (available on Payment Service API version 2.8 and above).

Field Name	Description
TRANS_VALUE	Requested transaction amount
CURR_CODE_ALPHA	Requested Currency (3-letter code)
TRANS_VALUE_SETTLED	Settled amount after currency conversion
CURR_CODE_ALPHA_SETTLED	Settled Currency (3-letter code)
TRANS_EXCH_RATE	Currency Conversion Rate (for more information, contact your gateway support representative)

16.3.1 Testing Multi-currency Response

An example of multi-currency response can be tested by using the Test Processor. Submitting a transaction to the test processor specifically with a EUR amount of 7.00 will yield an example of currency conversion in the response.

! Because this is a fictitious test scenario for demonstration purposes, the amounts in the response will always be the same, and *will not reflect* the actual EUR to USD conversion rate. This testing scenario will only work with the Test Processor.

16.3.2 Multi-Currency Response Example

In the response example below, the amount submitted was


```
{
: "REQUEST_ACTION":"CCAUTHORIZE",
: "REQ_ID":"4934041",
: "TRANS_STATUS_NAME":"APPROVED",
: "TRANS_VALUE":7,
: "CURR_CODE_ALPHA":"EUR",
: "TRANS_VALUE_SETTLED":"5.686415",
: "CURR_CODE_ALPHA_SETTLED":"USD",
: "TRANS_EXCH_RATE":0.812345,
: "TRANS_ID":2342342,
: "CUST_ID":242342235,
: "XTL_CUST_ID":"",
: "PO_ID":345345345,
: "XTL_ORDER_ID":"",
: "BATCH_ID":23423423,
: "PROC_NAME":"Test Processor",
: "MERCH_ACCT_ID":333333,
: "CARD_BRAND_NAME":"Visa",
: "CARD_TYPE":"",
: "CARD_PREPAID":"",
: "CARD_BANK":"",
```

```
: "CARD_BALANCE":"",
: "PMT_L4":"0002",
: "PMT_ID":5345453434,
: "PMT_ID_XTL":"",
: "PROC_UDF01":"",
: "PROC_UDF02":"",
: "PROC_AUTH_RESPONSE":"TEST23969",
: "PROC_RETRIEVAL_NUM":"7781C275-625C-4E10-A25651241742F7D8",
: "PROC_REFERENCE_NUM":"TEST97629125O",
: "PROC_REDIRECT_URL":"",
: "AVS_RESPONSE":"M",
: "CVV_RESPONSE":"M",
: "REQUEST_API_VERSION":"4.14",
: "P3DS_RESPONSE":"",
: "P3DS_VENDOR":"",
: "PO_LI_ID_1":"2342342",
: "PO_LI_COUNT_1":"1",
: "PO_LI_AMOUNT_1":"7.00",
: "PO_LI_PROD_ID_1":"22222",
: "MBSHP_ID_1":""
}
```


17 Partial Authorization

17.1 What is Partial Authorization?

Partial Authorization is a processor-specific feature that allows merchants to accept partial amount authorizations usually on prepaid and debit cards.

For inquiries regarding Partial Authorization support on your merchant account, contact your gateway support representative.

17.1.1 How It Works

- Merchant sends the Partial Authorization (PARTIAL_AUTH parameter) flag in their gateway transaction request, along with the minimum amount the merchant will accept for authorization (PARTIAL_AUTH_MIN parameter).
- Payment Service will attempt to authorize the full requested amount sent in LI_VALUE_1 parameter. If the requested minimum or full amount is available on the customer's card, the merchant will receive an "A*pproved*" transaction response.
  NOTE: The amount available for authorization will be returned in **TRANS_VALUE** gateway response field. Merchants must parse the value in this field in order to determine the actual authorization amount.

17.1.2 Partial Authorization Required Parameters

Field Name	Description
PARTIAL_AUTH	To enable Partial Authorization, set this parameter to "1" and send it with your **CCAUTHCAP** or **CCAUTHORIZE** gateway request. Set to "0" to disable or do not send the parameter at all.

PARTIAL_AUTH_MIN	This parameter sets the minimum amount that the merchant would accept for partial authorization. *Example: PARTIAL_AUTH_MIN=25.00 If a minimum of 25.00 is not available on the customer's card, the transaction request will be declined by the Payment Service.

17.1.3 Failed Transaction Due to Minimum Amount Response Example

```
<RESPONSE>
<REQUEST_ACTION>CCAUTHCAP</REQUEST_ACTION>
<TRANS_STATUS_NAME>DECLINED</TRANS_STATUS_NAME>
<TRANS_VALUE>21.00</TRANS_VALUE>
<TRANS_ID>989898</TRANS_ID>
<CUST_ID>101010</CUST_ID>
<XTL_CUST_ID/>
<MERCH_ACCT_ID>123</MERCH_ACCT_ID>
<CARD_BRAND_NAME>Visa</CARD_BRAND_NAME>
<CARD_DETAIL>DEBIT</CARD_DETAIL>
<CARD_CLASS>CONSUMER CREDIT</CARD_CLASS>
<CARD_COUNTRY>CRI</CARD_COUNTRY>
<CARD_TYPE>VISA CLASSIC</CARD_TYPE>
<CARD_PREPAID>1</CARD_PREPAID>
<CARD_BANK/>
<CARD_BALANCE/>
<PMT_L4>1111</PMT_L4>
<PMT_ID/>
<PMT_ID_XTL>
<API_RESPONSE>0</API_RESPONSE>
<API_ADVICE></API_ADVICE>
<SERVICE_RESPONSE>660</SERVICE_RESPONSE>
<SERVICE_ADVICE>Partial Approval</SERVICE_ADVICE>
<PROCESSOR_RESPONSE>10</PROCESSOR_RESPONSE>
<PROCESSOR_ADVICE>ApprvLesserAmt</PROCESSOR_ADVICE>
<INDUSTRY_RESPONSE>0</INDUSTRY_RESPONSE>
<INDUSTRY_ADVICE></INDUSTRY_ADVICE>
<REF_FIELD/>
<PROC_NAME>Processor Name</PROC_NAME>
<AVS_RESPONSE>U</AVS_RESPONSE>
<CVV_RESPONSE>M</CVV_RESPONSE>
<REQUEST_API_VERSION>4.14</REQUEST_API_VERSION>
<PO_LI_ID_1>878787</PO_LI_ID_1>
<PO_LI_COUNT_1>1</PO_LI_COUNT_1>
<PO_LI_AMOUNT_1>21.00</PO_LI_AMOUNT_1>
<PO_LI_PROD_ID_1>32320</PO_LI_PROD_ID_1>
</RESPONSE>
```

17.1.4 Partial Authorization Successful Response Example

```
<RESPONSE>
<REQUEST_ACTION>CCAUTHCAP</REQUEST_ACTION>
<TRANS_STATUS_NAME>APPROVED</TRANS_STATUS_NAME>
<TRANS_VALUE>25</TRANS_VALUE>
<TRANS_VALUE_SETTLED>25</TRANS_VALUE_SETTLED>
<CURR_CODE_ALPHA_SETTLED>USD</CURR_CODE_ALPHA_SETTLED>
```

```
<TRANS_EXCH_RATE/>
<TRANS_ID>989898</TRANS_ID>
<CUST_ID>1782177</CUST_ID>
<XTL_CUST_ID/>
<PO_ID>6889345</PO_ID>
<XTL_ORDER_ID>mytest1</XTL_ORDER_ID>
<BATCH_ID>12345</BATCH_ID>
<PROC_NAME>Processor Name</PROC_NAME>
<MERCH_ACCT_ID>123</MERCH_ACCT_ID>
<CARD_BRAND_NAME>Visa</CARD_BRAND_NAME>
<CARD_TYPE>VISA CLASSIC</CARD_TYPE>
<CARD_DETAIL>CREDIT</CARD_DETAIL>
<CARD_CLASS>CONSUMER CREDIT</CARD_CLASS>
<CARD_COUNTRY>CRI</CARD_COUNTRY>
<CARD_PREPAID>1</CARD_PREPAID>
<CARD_BANK/>
<CARD_BALANCE/>
<PMT_L4>1111</PMT_L4>
<PROC_UDF01/>
<PROC_UDF02/>
<PROC_AUTH_RESPONSE>XXYYXX123</PROC_AUTH_RESPONSE>
<PROC_RETRIEVAL_NUM>083073744911846</PROC_RETRIEVAL_NUM>
<PROC_REFERENCE_NUM/>
<AVS_RESPONSE>U</AVS_RESPONSE>
<CVV_RESPONSE>M</CVV_RESPONSE>
<REQUEST_API_VERSION>4.14</REQUEST_API_VERSION>
<PO_LI_ID_1>455555</PO_LI_ID_1>
<PO_LI_COUNT_1>1</PO_LI_COUNT_1>
<PO_LI_AMOUNT_1>25</PO_LI_AMOUNT_1>
<PO_LI_PROD_ID_1>77770</PO_LI_PROD_ID_1>
<MBSHP_ID_1/>
</RESPONSE>
```

18 External Order ID Uniqueness

18.1 UNIQUE_XTL_ORDER_ID Parameter

This parameter allows the merchant to control how the Payment Service handles External Order ID's (XTL_ORDER_ID) uniqueness the payment system. This parameter should only be sent with CCAUTHCAP or CCAUTHORIZE requests.

For inquiries regarding this parameter, contact your gateway support representative.

18.1.1 UNIQUE_XTL_ORDER_ID Setting

Value	Description
0	Send this value (or do not send the parameter at all) to disable XTL_ORDER_ID uniqueness enforcement. Default behavior.

1	Send this value to enable XTL_ORDER_ID uniqueness enforcement; this will also tell the gateway to decline the request. If the XTL_ORDER_ID value already exists in a previously **approved** order, the gateway is going to decline the transaction request.
2	Send this value to enable XTL_ORDER_ID uniqueness enforcement; if the XTL_ORDER_ID value already exists in a previously **approved** order, the gateway is going to return the Service Response of the matching order/transaction.

18.1.2 Example Decline Response when UNIQUE_XTL_ORDER_ID parameter is set to "1":

```
<RESPONSE>
  <REQUEST_ACTION>CCAUTHORIZE</REQUEST_ACTION>
  <TRANS_STATUS_NAME/>
  <TRANS_VALUE/>
  <TRANS_ID/>
  <CUST_ID/>
  <XTL_CUST_ID>1234567MMBBR</XTL_CUST_ID>
  <MERCH_ACCT_ID>100</MERCH_ACCT_ID>
  <CARD_BRAND_NAME/>
  <PMT_L4/>
  <PMT_ID/>
  <PMT_ID_XTL/>
  <API_RESPONSE>0</API_RESPONSE>
  <API_ADVICE></API_ADVICE>
  <SERVICE_RESPONSE>685</SERVICE_RESPONSE>
  <SERVICE_ADVICE>Duplicate Order Detected</SERVICE_ADVICE>
  <PROCESSOR_RESPONSE>0</PROCESSOR_RESPONSE>
  <PROCESSOR_ADVICE></PROCESSOR_ADVICE>
  <INDUSTRY_RESPONSE>0</INDUSTRY_RESPONSE>
  <INDUSTRY_ADVICE></INDUSTRY_ADVICE>
  <REF_FIELD>XTL_ORDER_ID</REF_FIELD>
  <PROC_NAME/>
  <AVS_RESPONSE/>
  <CVV_RESPONSE/>
  <REQUEST_API_VERSION>4.14</REQUEST_API_VERSION>
</RESPONSE>
```

19 Processor Specific Fields

19.1 Processor Specific Fields

These parameters can be used by merchants to send processor-specific data in their gateway transaction request.

Field Name	Used For

PROC_UDF01	eMerchantPay: **ThreatMetrix™** eMerchantPay will occasionally require merchants to add ThreatMetrix™ THM_ID to the transaction data. Use the PROC_UDF01 parameter in all initial CCAUTHCAP and CCAUTHORIZE requests for this purpose. Other processor-specific data.
PROC_UDF02	Reserved for NATS Affiliate Software.

For more information regarding these fields, contact your gateway support representative.

20 PagoEfectivo

Pago Efectivo is an online payment method in Peru that allows online shoppers to pay through online banking or in cash at authorized centers. Customers are able to check out with a unique payment identification code – **CIP** (Código de identificación de pago) instead of a credit or debit card. Where supported, a QR Code will be generated and the Customer will be able to scan and pay through their bank or copy the QR Code URL and pay through a computer browser.

20.1 PagoEfectivo Features

Type of Payment	*Offline*
Chargeback	*No*
Consumer Currency	*PEN (USD supported)*
Consumer Country	*Peru*
Expiry Period	*2.5 hrs*
Recurring Payment	*No*
Partial Payment Support	*No*
Refunds	*No (Check w/Merchant)*

20.2 PagoEfectivo Request Parameters

The below parameters are required to make a successful request for processing a PagoEfectivo sale.

Field Name	Description
request_action	**PAGSALE**
li_prod_id_x	Line Item Count Max value is “10”
li_value_x	Line Item Transaction Amount
userDocumentType	Customer ID Type
userDocumentNumber	Customer Document number (Max 14)

merch_acct_id	Merchant Account ID
cust_fname	Customer's First Name
cust_lname	Customer's Last Name
bill_addr	Customer Billing Street Address
bill_addr_city	Customer's Billing City
bill_addr_state	Customer's Billing State
bill_addr_zip	Customer's Billing Postal/ZIP code
bill_addr_country	Customer's Billing Country
cust_email	Customer's Email Address
request_currency	3-letter Currency Code. Ex: PEN

20.2.1 userDocumentType Explained

Value for userDocumentType	Description
DNI	National Identity Document
PAR	Departure
PAS	Passport
LMI	Military Notebook
NAN	Other

20.3 Additional Response Parameters

Field Name	Description
Proc_redirect_url	The address to which a Customer is redirected once the QR Code has been generated
proc_retrieval_num	Processor tracking number
Proc_reference_num	Contains CIP number
dateExpiry	CIP Expiry Date

20.4 PagoEfectivo Request Example

```
https://api.inoviopay.com/payment/pmt_service.cfm
?site_id=328776&li_prod_id_1=183238&cust_fname=customer&cust_lname=Tester&bill_addr_state=CA&request_response_format=JSON&request_api_version=4.14&cust_email=test@test.com&bill_addr_zip=90401&bill_addr_country=US&request_action=PAGSALE&req_username=testing@example&req_password=Maxell0022.@&bill_addr_city=Los Angeles&bill_addr=1 MainSt&userDocumentType=NAN&merch_acct_id=157923&userDocumentNumber=1234567891234456&li_value_1=15&request_currency=USD&cust_phone=+1 202-918-213212345
```


```
{
 "site_id":"328776",
 "li_prod_id_1":"183238",
 "cust_fname":"customer",
 "cust_lname":"Tester",
 "bill_addr_state":"CA",
 "request_response_format":"JSON",
 "request_api_version":"4.14",
 "cust_email":"test@test.com",
 "bill_addr_zip":"90401",
 "bill_addr_country":"US",
 "request_action":"PAGSALE",
 "req_username":"testero@example.com",
 "req_password":"Maxell0022.#",
 "bill_addr_city":"Los Angeles",
 "bill_addr":"1 Main St",
 "userDocumentType":"DNI",
 "merch_acct_id":"050023",
 "userDocumentNumber":"1234567891234456",
 "li_value_1":"60",
 "request_currency":"PEN",
 "cust_phone":" 1 202-918-213212345"
}
```


20.5 PagoEfectivo Response Example

Below is a sample response for a PAGSALE request. The `PROC_REDIRECT_URL` is the URL to display the CIP and QR Code for further payment.


```
{
  "REQUEST_ACTION": "PAGSALE",
  "REQ_ID": "58931022114",
  "TRANS_STATUS_NAME": "PENDING",
  "TRANS_VALUE": 60,
  "CURR_CODE_ALPHA": "PEN",
  "TRANS_VALUE_SETTLED": 60,
  "CURR_CODE_ALPHA_SETTLED": "PEN",
  "TRANS_EXCH_RATE": "",
  "TRANS_ID": 3943973448,
  "CUST_ID": 361302333,
  "XTL_CUST_ID": "",
  "PO_ID": 3887678760,
  "XTL_ORDER_ID": "",
  "BATCH_ID": 7146505,
  "PROC_NAME": "PagoEffectivo",
  "MERCH_ACCT_ID": 157923,
  "CARD_BRAND_NAME": "PagoEfectivo",
```

```
  "CARD_TYPE": "",
  "CARD_PREPAID": "",
  "CARD_BANK": "",
  "CARD_BALANCE": "",
  "CARD_DETAIL": "",
  "PMT_L4": "2345",
  "PMT_ID": 487448490,
  "PMT_ID_XTL": "",
  "PMT_AAU_UPDATE_DT": "",
  "PMT_AAU_UPDATE_DESC": "",
  "PROC_UDF01": "CA",
  "PROC_UDF02": "",
  "PROC_AUTH_RESPONSE": "",
  "PROC_RETRIEVAL_NUM": "",
  "PROC_REFERENCE_NUM": "161680789",
  "PROC_BARCODE": "",
  "PROC_REDIRECT_URL": "https://payment.pagoefectivo.pe/C800BC64-47D0-4EE8-BF1F-
7B36D8206524.html",
  "AVS_RESPONSE": "",
  "CVV_RESPONSE": "",
  "REQUEST_API_VERSION": "4.14",
  "P3DS_VENDOR": "",
  "P3DS_RESPONSE": "",
  "PO_LI_ID_1": "2658110558",
  "PO_LI_COUNT_1": 1,
  "PO_LI_AMOUNT_1": "60",
  "PO_LI_PROD_ID_1": "183238",
  "MBSHP_ID_1": ""
}
```


21 Apple Pay

21.1 Overview

This section discusses how to send credit card transactions to the Payment Gateway using Apple Pay. In order to do so, the web pages which host the payment forms must have their domains registered. See the document *Apple Pay Registration Process* for more details.

Additionally, the Processor and Merchant Account being used must also support Apple Pay. For more information on technical usage, please contact your gateway support representative.

21.2 Customers who can use Apple Pay

Customers who are using Apple devices (iPhone, iPad, MacBook, etc.) and Apple browser (e.g. Safari) are able to use Apple Pay to make purchases. This section discusses how to set up an Apple Pay purchase on your hosted Payment Page, and then send that as a transaction to the gateway API.

21.3 Flow

This is a list of the order in which you can set up the things needed to process transactions with Apple Pay.

- Fetch the Apple Pay configuration data from the gateway
- Update the configuration with the specific payment/order details
- Present the Apple Pay button
- Add the Apple Pay library to the payment page
- Create a JS function to validate your domain
- Create and start the Apple Pay session
- Get the authorized payment information from the customer's browser
- Send the transaction to the gateway

Each item in the list is discussed in the following sections, and some example code is provided. Note that all code provided is for use as an example only, and does not represent code that can be used in your production environment.

NOTE: Real cards must be used when testing in production environment. Test cards will not work.

21.4 Fetch Apple Pay configuration data from the gateway

You will need to get configuration data from the gateway in order to set up the Apple Pay session when the customer clicks the Apple Pay button. This comes from a request to the following endpoint:

Endpoint: https://api.inoviopay.com/payment/applepay.cfm

- **Requests must be made in JSON format**

Parameters:

Field Name	Description
REQ_USERNAME	API CREDENTIAL USERNAME
REQ_PASSWORD	API CREDENTIAL PASSWORD
DOMAIN_NAME	DOMAIN NAME WHERE THE PAYMENT PAGE IS HOSTED/SERVED FROM
CLIENT_ID	CLIENT ID
REQUEST_ACTION	REQUEST_ACTION = APPLEPAYCONFIG USED TO INSTRUCT THE ENDPOINT TO PROVIDE THE APPLE PAY CONFIGURATION

Note: All parameters are required

The response will be a JSON-formatted string. Here is an example of the structure:

```
{
 "APPLEPAY_CLI_CONF_ID": xx, // internal unique id; do not change
 "REQ_ID": 11111111111, // trace ID, used for troubleshooting
 "CLIENT_ID": 111111, // your gateway api client_id
 "INITIATIVE": "web", // identifies e-commerce transaction
 "DISPLAY_NAME": "xxxxxxx", // Short, localized description of the merchant.
 "PARTNER_MERCH_NAME": "xxxxxx", //name of the merchant
```

```
 "PARTNER_INTERNAL_MERCH_ID": "xxxxxxxxxxxxx", // identifies the merchant to apple
 "ENCRYPT_TO": "xxxxxxxxxxxxxxxxxxxx", // merchant ID to Apple
 "DOMAIN_LIST": [ // domain registered in Inovio  portal serving this page
  {
   "DOMAIN_NAME": "paymentpagedomain.com",
   "DOMAIN_STATUS": "ACTIVE"
  }
 ],
 "paymentRequest": { //
  "countryCode": "US",
  "currencyCode": "USD",
  "merchantCapabilities": [
   "supports3DS"
  ],
  "supportedNetworks": [
   "visa",
   "masterCard"
  ],
  "requiredBillingContactFields": [
   "postalAddress",
   "name",
   "phoneticName",
   "phone",
   "email"
  "total": {
   "label": "" // required; a short, localized description of the line item
   "type": "final", // do not change
   "amount": 0 // must be greater than or equal to zero       }
  },
 },
 "fetchUrl": "https://api.inoviopay.com/apple-pay-services/api/session/create"
}
```

This should be provided on your payment page in the parameter: merchantConfig

Example:

```
<script>
 this.merchantConfig = {
  "APPLEPAY_CLI_CONF_ID": 11,
  "REQ_ID": 11111111111,
  "CLIENT_ID": 1111111,
  "INITIATIVE": "web",
  "DISPLAY_NAME": "xxxxxxx",
  "PARTNER_MERCH_NAME": "xxxxxx",
  "PARTNER_INTERNAL_MERCH_ID": "xxxxxxxxxxxxx",
  "ENCRYPT_TO": "xxxxxxxxxxxxxxxxxxxx",
  "DOMAIN_LIST": [
   {
    "DOMAIN_NAME": "paymentpagedomain.com",
    "DOMAIN_STATUS": "ACTIVE"
   }
  ],
```

```
  "paymentRequest": {
   "countryCode": "US",
   "currencyCode": "USD",
   "merchantCapabilities": [
    "supports3DS"
   ],
   "supportedNetworks": [
    "visa",
    "masterCard"
   ],
   "requiredBillingContactFields": [
    "postalAddress",
    "name",
    "phoneticName",
    "phone",
    "email"
   "total": {
    "label": ""
    "type": "final",
    "amount": 0
   },
  },
  "fetchUrl": "https://api.inoviopay.com/apple-pay-services/api/session/create"
 }
</script>
```

21.5 Update the configuration with the specific payment/order details

Once you have the configuration stored in the merchantConfig parameter on your payment page, you can update the details which are specific to the purchase.

Example:

```
<script>
  merchantConfig.total.label = “Some purchase description total”;
  merchantConfig.total.amount = 49.95
</script>
```

21.6 Add the Apple Pay library to your payment page

You must add the Apple Pay library to your payment page to make calls to the Apple Pay API. To do that, source the Apple Pay library script scource in the page’s source html

Example:

```
<script src="https://applepay.cdn-apple.com/jsapi/v1/apple-pay-sdk.js"></script>
```

Note that this library will include functionality to validate the user/browser/device as eligible for Apple Pay.

21.7 Create a JS function to validate your domain

The Apple Pay JS API library will need to call a function that you define in order to perform a validation on the domain where your payment page is hosted. Note that all of this must load directly from your servers on this domain (no proxies, iframes, etc.)

Example:

```
<script>
 function validateMerchant(merchantConfig) {
  console.log("validateMerchant: ", merchantConfig.DOMAIN_LIST)
  const data={
   method: "POST",
   headers: {
    "Content-type": "application/json",
    "Accept": "*/*"
   },
   body: JSON.stringify({
    "merchantIdentifier": merchantConfig.PARTNER_INTERNAL_MERCH_ID,
    "displayName": merchantConfig.DISPLAYNAME,
    "initiative": merchantConfig.INITIATIVE,
    "initiativeContext": merchantConfig.DOMAIN_LIST[0].DOMAIN_NAME
   })
  };
  return fetch(merchantConfig.fetchUrl,data);
 }
<script>
```

21.8 Present the Apple Pay button

Once the Apple Pay library is added, you can add an HTML element. Please note that it is required to name the element “apple-pay-button”.

Example:

```
<style>
.apple-pay-button {
  --apple-pay-button-width: 150px;
  --apple-pay-button-height: 30px;
  --apple-pay-button-border-radius: 3px;
  --apple-pay-button-padding: 0px 0px;
  --apple-pay-button-box-sizing: border-box;
}
</style>
```

The customer will see the Apple Pay button:

21.9 Create and start the Apple Pay session

After presenting the button, you must add an action to it which will start up the session where the browser will pop up a dialog for the user to select their card and approve the transaction.

For this, add an onClick event to the Apple Pay button. When referring to "paymentRequest", we are looking at the JSON object returned from APPLEPAYCONFIG. An example of this is included below

21.9.1 Handle the Authorized Payment

After the customer has authorized the payment via the Apple Pay overlay, Safari will return an object (Apple Pay Token) with the data needed to transfer to the gateway API to authorize a payment.

For this, you must create the session.onpaymentauthorized method to receive the Apple Pay token object. An example of this is included below

21.9.2 Handle a Canceled Payment or Error

You should also handle the case where the customer does not authorize the payment or in case an error is thrown. For this, you must create the session.onCancel. An example of this is included below. Here is an example of all of these elements described above:

```
<script>
 async function onApplePayClick() {
  // Check for valid ApplePay session
  if(!ApplePaySession){
   return;
  }
  // Get the paymentRequest data from the merchantConfig object
  const paymentRequest = this.merchantConfig.paymentRequest;
  // Validate the merchant.
  // Note that this.merchantConfig is defined in Step 3
  const session = new ApplePaySession(3, paymentRequest);
  session.onvalidatemerchant = async (event) => {
   const merchantSession = await
    this.validateMerchant(this.merchantConfig).then(res=>res.json());
   session.completeMerchantValidation(merchantSession);
  };
    // Handle the authorized payment
  // Note, this must come inside the onApplePayClick() function, and
  // before session.begin
  session.onpaymentauthorized = async (event) => {
   // Define ApplePayPaymentAuthorizationResult
   const result = {
    status: ApplePaySession.STATUS_SUCCESS,
   };
   var applePaymentToken = event.payment;
   var tokenEncoded = btoa(JSON.stringify(applePaymentToken));
   // Now that the payment is authorized in Apple Pay, you can
   // initiate a service call to the gateway passing the
   // applePaymentToken into the gateway API parameter: PMT_WALLET_CRYPTOGRAM
```

```
    //
    // Do that here, through your server

    // If the payment is declined at the processor, then you should
    // put an appropriate message in place to the customer
    // result.status = ApplePaySession.STATUS_FAILURE

    // Now we can gracefully complete the Apple Pay browser interaction
    session.completePayment(result);
  };
  // In case the customer cancels or in case of an unexpected error
  // Note, this must come inside the onApplePayClick() function, and
  // before session.begin
  session.oncancel = (event) => {
    // Payment canceled by WebKit
    // error handling
    console.log("Payment canceled by WebKit: "+JSON.stringify(event.error))
  }
  // now the user will interact to approve the purchase
  // Note, this must come at the END of the onApplePayClick()
  session.begin();
 }
</script>
```

21.10 Send the authorized payment data from the session to the gateway

Here is an example of a request to authorize payment with Apple Pay:

```
https://api.inoviopay.com/payment/pmt_service.cfm  ?request_action=CCAUTHCAP&
li_count_1=1&li_prod_id_1=111&li_value_1=49.95req_username=GATEWAY
USER&req_password=GATEWAY
PASS&site_id=11111&request_response_format=JSON&request_api_version=4.14&request_c
urrency=USD&PMT_WALLET=applepay &PMT_WALLET_CRYPTOGRAM=xxxxxxx
```

21.11 Important requirements for Apple Pay authorization requests

You must NOT include any of the following parameters, which will cause the gateway to reject your request:

```
PMT_NUMB
PMT_KEY
PMT_EXPIRY
TOKEN_GUID
PMT_ID
PMT_LAST4
PMT_ID_XTL
PMT_NUMB_COF
REQUEST_INITIATOR
```

Other gateway parameters can be include, but MUST match the values used in the Apple Pay session.

- Parameter li_value_1 must match the value of paymentRequest.total in the merchantConfig data.
- Parameter request_currency must match the value of paymentRequest.currencyCode in the merchantConfig data

21.12 Rebiling and Credentials on File

The Inovio response will return a CUST_ID and PMT_ID on a successful authorization using an Apple Pay token (in the same way that it does for normal transactions.

To authorize on the generated ApplePay payment record, the same CUST_ID and PMT_ID should be passed in on all subsequent rebills or unscheduled merchant initiated card-on-file transaction requests.

22 Google Pay

22.1 Overview

This section discusses how to send credit card transactions to the Payment Gateway using Google Pay. In order to do so, the web pages which host the payment forms must have their domains registered. **Additionally, the Processor and Merchant Account being used must also support Google Pay.** Please contact your gateway support representative for more information.
Click here for an overview of Google Pay, Google Pay Integration Checklist, and Google Pay Branding Guidelines. For more information on Google Pay:

- List of payment methods that support Google Pay. Your list of allowedCardNetworks will be determined by your merchant account configuration.
- List of countries (regions) where you can use Google Pay. The Gateway supports the currencies enabled by your merchant account configuration.
- List of supported browsers.

22.2 Implementation Process Flow

Below are the necessary steps to utilize Google Pay properly within the Payment Gateway. Each item in the list is discussed in the following sections. Some example code is provided. Note that all code provided is for use as an example only, and does not represent code that can be used in your production environment.

- Fetch your Google Pay configuration data from the gateway
- Add the Google Pay library to your payment page
- Customizing the Google Pay Button
- Handling the Google Pay Loaded Event
- Presenting the Google Pay Button
- Handling the Google Pay Button Click
- Payment Authorization
- Submitting authorized payment data

NOTE: Real PAN must be used when testing in a production environment. Test cards will not work.

22.3 Fetch your Google Pay configuration data from the gateway

You will need to get your configuration data from the Payment Gateway in order to set up the Google Pay session after the customer clicks the Google Pay button. This comes from a request to the following endpoint:
Endpoint: **https://api.inoviopay.com/payment/googlepay.cfm**

- **Requests must be made in JSON format.**

Parameters:

Field Name	Description
REQ_USERNAME	API CREDENTIAL USERNAME
REQ_PASSWORD	API CREDENTIAL PASSWORD
DOMAIN_NAME	DOMAIN NAME WHERE THE PAYMENT PAGE IS HOSTED/SERVED FROM
CLIENT_ID	CLIENT ID
REQUEST_ACTION	REQUEST_ACTION = GOOGLEPAYCONFIG Used to instruct the endpoint to provide the proper Google Pay Configuration for your unique information

Note: All parameters are required

Sample Body Request

```
{
"request_login": "googlepayClient@Inovio.com",
"request_password": "Test123",
"request_action": "GOOGLEPAYCONFIG",
"domain_name": "paymentpagedomain.com",
"client_id": "100"
}
```

Sample Response

The response will be a JSON-formatted string.

```
{
    "GOOGLEPAY_CLI_CONF_ID": 1,
    "REQ_ID": 2744,
    "CLIENT_ID": 100,
    "DOMAIN_LIST": [
        {
            "DOMAIN_NAME": "paymentpagedomain.com",
            "DOMAIN_STATUS": "ACTIVE"
        }
    ],
    "hostConfig": {
        "merchantId": "BC...JW",
        "environment": "TEST",
        "merchantName": "Inovio",
        "gatewayId": "inoviopay",
        "gatewayMerchantId": "string-string"
    }
}
```

This should be provided on your payment page in the parameter: merchantConfig.

* Google Pay field

22.4 Add the Google Pay library to your payment page

Import the JavaScript library into your payment page. Then, add an empty DIV element titled "gpay-container". This DIV acts as a placeholder where the Google Pay button will appear. You can place this DIV anywhere you want the button to show up on your website.

```
<!DOCTYPE html>
<html lang="en">
 <script>
   function onGPayApiLoaded() {
     googlePayHandler.initialize({
       gpayButtonContainerId: 'gpay-container' // The ID of the div where the button
should appear
     });
```

```
  }
 </script>
  <body>
    <div id="gpay-container"></div>
    <script type="text/javascript" src="main.js"></script>
    <script async src="https://pay.google.com/gp/p/js/pay.js"
onload="onGPayApiLoaded()"></script>
  </body>
</html>
```

The is your designated spot for the button.
The script tag for pay.js asynchronously loads the Google Pay library.
The onload="onGooglePayLoaded()" attribute ensures that your onGooglePayLoaded() function (which checks if Google Pay is ready) runs as soon as the library finishes loading.

22.5 Customizing the Google Pay button

The Google Pay button should fit with your website's design and user experience. Consult the documentation from Google for how to achieve this goal. After the Google Pay API is loaded and ready, your site can display the Google Pay button. It is dynamically generated by the Google Pay library and placed inside the gpay-container you made.

```
const GPAY_BUTTON_CONTAINER_ID = 'gpay-container';
function renderGooglePayButton() {
  const button = getGooglePaymentsClient().createButton({
    buttonColor: 'default',
    buttonType: 'buy',
    buttonRadius: 4,
    buttonLocale: 'en',
    onClick: onGooglePaymentButtonClicked,
    allowedPaymentMethods: baseGooglePayRequest.allowedPaymentMethods,
  });
  document.getElementById(GPAY_BUTTON_CONTAINER_ID).appendChild(button);
}
```

The createButton() library method takes the ButtonOptions configuration argument that lets you define how you want the button to look and behave.

GPAY_BUTTON_CONTAINER_ID: Constant to hold the ID (gpay-container) of the HTML element where you want the button to appear.
renderGooglePayButton(): Function that is responsible for creating and adding the button.
getGooglePaymentsClient().createButton({...}): Uses the Google Pay client to generate the button. The ButtonOptions object then allows for customization:

- buttonColor: Choose the appearance (e.g., 'default', 'black', 'white').
- buttonType: Defines the text on the button (e.g., "Buy/Checkout with Google Pay").
- buttonRadius: Adjusts the roundness of the button's corners.
- buttonLocale: Sets the language for the button's text.
- onClick: onGooglePaymentButtonClicked: Prompts the button to call your onGooglePaymentButtonClicked function (which initiates the payment process) whenever a customer clicks on it.
- allowedPaymentMethods: Ensures the button only appears if there are payment methods available that match what you've configured in your baseGooglePayRequest.

document.getElementById(...).appendChild(button): Takes the button that Google Pay created and inserts it directly into the DIV element on your webpage, making it visible to the user.

22.6 Handling the Google Pay Loaded Event

The onGooglePayLoaded() function is called when the Google Pay API script has finished loading. In essence, onGooglePayLoaded() is the handshake with the Google Pay service, ensuring everything is in order before you present the payment option to your customers.

```
function onGooglePayLoaded() {
  const req = deepCopy(baseGooglePayRequest);
  getGooglePaymentsClient()
    .isReadyToPay(req)
    .then(function (res) {
      if (res.result) {
        renderGooglePayButton();
      } else {
        console.log('Google Pay is not ready for this user.');
      }
    })
    .catch(console.error);
}
```

22.7 Configuring Google Pay

The baseGooglePayRequest object defines the fundamental configuration for all Google Pay requests.

```
const baseGooglePayRequest = {
  apiVersion: 2,
  apiVersionMinor: 0,
  allowedPaymentMethods: [
    {
      type: 'CARD',
      parameters: {
        allowedAuthMethods: ['PAN_ONLY', 'CRYPTOGRAM_3DS'],
        allowedCardNetworks: ['AMEX', 'DISCOVER', 'MASTERCARD', 'VISA'],
      },
      tokenizationSpecification: {
        type: "PAYMENT_GATEWAY",
        parameters: {
          "gateway": "inoviopay",
          "gatewayMerchantId": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
                  }
      },
    },
  ],
  merchantInfo,
};
Object.freeze(baseGooglePayRequest);
let paymentsClient = null;
function getGooglePaymentsClient() {
  if (paymentsClient === null) {
    paymentsClient = new google.payments.api.PaymentsClient({
      environment: 'TEST', // Environment configuration
      merchantInfo,
      paymentDataCallbacks: {
        onPaymentAuthorized: onPaymentAuthorized,
        onPaymentDataChanged: onPaymentDataChanged,
      },
    });
  }
  return paymentsClient;
}
```

Key properties that shape the Google Pay experience:

- apiVersion and apiVersionMinor: These simply tell Google Pay which version of the Google Pay API you are using.
- allowedPaymentMethods: Array that declares what types of payments you accept.
    - parameters: Within the CARD type, you specify the details:
        - allowedAuthMethods: Defines the authentication methods your integration supports:
            - PAN_ONLY: Used for cards saved directly to a user's Google account. When used, Google Pay returns the actual card number.

        - To use 3D Secure with PAN_ONLY, submit your third party 3-D Secure details with the authorization.
        - The Payment Gateway will not prevent a transaction from being sent to the processor without 3DS on the transaction request.
      - CRYPTOGRAM_3DS: Used cards tokenized via the Google Wallet app. Instead of the real card number, a device-specific token is used. A 3-D Secure cryptogram is generated on the user's device, providing stronger security and often shifting liability for fraud to the card issuer. We recommend using this allowedAuthMethods only.
    - allowedCardNetworks: List all the major card networks your Merchant Account is set up to accept: American Express, Discover, Mastercard, Visa, and/or more. For more information, see the official Google Pay documentation.
  - tokenizationSpecification: Configuration for how payment credentials are secured and sent to your payment processor.
    - type: "PAYMENT_GATEWAY": Indicates that the payment gateway will handle the tokenization process on your behalf.
    - parameters: These are gateway-specific details essential for secure processing provided to you when you make a request for your configuration data from the gateway:
      - gateway: The name of your payment gateway (e.g., "inoviopay").
      - gatewayMerchantId: Your unique ID provided by the gateway.
- Object.freeze(baseGooglePayRequest): Prevents any accidental changes to your base Google Pay configuration after it is set.
- The paymentsClient variable is initially null. The getGooglePaymentsClient() function uses a common pattern called "lazy initialization." It creates the PaymentsClient instance only when ready, improving performance by not creating it prematurely.
- environment: The above example is set to 'TEST', which is perfect for development and debugging. Remember to change this to 'PRODUCTION' when you're ready to go live with real transactions!
- paymentDataCallbacks: Define the functions to handle events during the Google Pay flow:
  - onPaymentAuthorized: Function to be called after the user has successfully authorized a payment. This is where you'd typically send the payment data to the payment service provider for processing.
  - onPaymentDataChanged: Callback that fires if the user makes changes to their payment data (like selecting a different shipping address or payment method) within the Google Pay sheet, allowing you to dynamically update the order total or shipping options.

22.8 Handling the Google Pay Button Click

When a user clicks your Google Pay button, the onGooglePaymentButtonClicked() function swings into action. Its job is to gather all the specifics for the current transaction and then launch the Google Pay payment window.

```
// @namespace googlePayHandler
// @description A self-contained handler for a professional and secure Google Pay integration.
// This script manages fetching configuration from a secure backend, setting up the Google Pay client, and handling the entire payment lifecycle.
const googlePayHandler = {
  // Default settings can be overridden by the customer during initialization.
  config: {
    gpayButtonContainerId: 'gpay-container',
```

```
    currencyCode: 'USD',
    countryCode: 'US',
    environment: 'TEST', // Should be 'PRODUCTION' for live transactions
  },
  // --- Internal State ---
  paymentsClient: null,
  merchantConfig: null,
  // CRITICAL: Fetches the merchant configuration from the CUSTOMER'S backend.
  // This is a vital security measure. API credentials should NEVER be exposed in
client-side code.
  // Your customer must implement a server endpoint that securely communicates with
your API.
  // @returns {Promise<Object|null>} A promise that resolves with the merchant
configuration or null on failure.
  async fetchMerchantConfigFromServer() {
    // --- CUSTOMER ACTION REQUIRED ---
    // This URL must point to an endpoint on YOUR CUSTOMER'S server.
    // Their server is responsible for making the secure, server-to-server call to the
Inovio API.
    const customerBackendUrl = 'https://api.customer-website.com/get-payment-config';
    try {
      const response = await fetch(customerBackendUrl);
      if (!response.ok) {
        throw new Error(`Network response was not ok: ${response.statusText}`);
      }
      const configData = await response.json();
      console.log("Successfully fetched merchant configuration.");
      return configData;
    } catch (error) {
      console.error("Fatal Error: Could not retrieve merchant configuration from the
server.", error);
      // Optional: Display a user-friendly error message in the UI.
      // e.g., document.getElementById(this.config.gpayButtonContainerId).innerText =
"Payment system unavailable.";
      return null;
    }
  },
  // Initializes the Google Pay client if it doesn't already exist.
  // @returns {google.payments.api.PaymentsClient}
  getGooglePaymentsClient() {
    if (this.paymentsClient === null) {
      if (!this.merchantConfig) {
        throw new Error("Cannot initialize Google Payments client without merchant
configuration.");
      }
      this.paymentsClient = new google.payments.api.PaymentsClient({
        environment: this.config.environment,
        merchantInfo: {
            merchantId: this.merchantConfig.HOST_CONFIG.merchantId,
            // Customer will set their own display name
            merchantName: this.merchantConfig.HOST_CONFIG.merchantName || 'Sample
Merchant',
        },
        paymentDataCallbacks: {
          onPaymentAuthorized: this.onPaymentAuthorized,
          onPaymentDataChanged: this.onPaymentDataChanged,
```

```
        },
      });
    }
    return this.paymentsClient;
  },
  // Main entry point. Initializes the Google Pay flow.
  // @param {Object} [userConfig={}] - Customer-specific configuration to override the
defaults.
  async initialize(userConfig = {}) {
    // 1. Merges customer's configuration with defaults.
    this.config = { ...this.config, ...userConfig };
    // 2. Securely fetches the configuration from the customer's backend.
    this.merchantConfig = await this.fetchMerchantConfigFromServer();
    if (!this.merchantConfig) {
      console.error("Google Pay initialization failed: Merchant configuration is
missing or could not be fetched.");
      return;
    }
      const isReadyToPayRequest = this.createBaseRequest();
   this.getGooglePaymentsClient()
      .isReadyToPay(isReadyToPayRequest)
      .then((isReadyToPayRequest) => {
        if (isReadyToPayRequest.result) {
          this.renderGooglePayButton();
        } else {
          console.log('Google Pay is not ready for this user.');
        }
      })
      .catch(console.error);
  },
  // Creates and appends the Google Pay button to the configured container.
  renderGooglePayButton() {
    const container = document.getElementById(this.config.gpayButtonContainerId);
    if (!container) {
        console.error(`Google Pay button container with ID
"${this.config.gpayButtonContainerId}" was not found in the DOM.`);
        return;
    }
    // Clears any previous content (e.g. error messages)
    container.innerHTML = '';
    const button = this.getGooglePaymentsClient().createButton({
      onClick: this.onGooglePaymentButtonClicked.bind(this),
      allowedPaymentMethods: this.createBaseRequest().allowedPaymentMethods,
    });
    container.appendChild(button);
  },
  // Creates the base payment request object required by the Google Pay API.
  // @returns {Object} The Google Pay base request object.
  createBaseRequest() {
    return {
      apiVersion: 2,
      apiVersionMinor: 0,
      allowedPaymentMethods: [{
        type: 'CARD',
        parameters: {
          allowedAuthMethods: ['PAN_ONLY', 'CRYPTOGRAM_3DS'],
```

```
          allowedCardNetworks: ['AMEX', 'DISCOVER', 'INTERAC', 'JCB', 'MASTERCARD',
'VISA'],
        },
        tokenizationSpecification: {
          type: 'PAYMENT_GATEWAY',
          parameters: {
            gateway: this.merchantConfig.HOST_CONFIG.gatewayId,
            gatewayMerchantId: this.merchantConfig.MERCH_IDENTIFIER,
            // These values should come from your secure config fetch
          },
        },
      }],
    };
  },
  // Handles the click event from the Google Pay button.
  onGooglePaymentButtonClicked() {
    const paymentDataRequest = {
      ...this.createBaseRequest(),
      transactionInfo: {
        countryCode: this.config.countryCode,
        currencyCode: this.config.currencyCode,
        totalPriceStatus: 'FINAL',
        totalPrice: '10.00',
      },
      merchantInfo: {
        merchantId: this.merchantConfig.MERCH_IDENTIFIER,
        merchantName: this.merchantConfig.MERCHANT_NAME || 'Sample Merchant'
      },
      // The callbackIntents must match the callbacks provided in
getGooglePaymentsClient().
      // We provided onPaymentAuthorized and onPaymentDataChanged.
      // To handle dynamic shipping, you would add 'SHIPPING_ADDRESS' and
'SHIPPING_OPTION' and then build out the logic in the onPaymentDataChanged callback.
      callbackIntents: ['PAYMENT_AUTHORIZATION'],
    };
    console.log('Requesting payment data...', paymentDataRequest);
    this.getGooglePaymentsClient()
      .loadPaymentData(paymentDataRequest)
      .catch(err => console.error("loadPaymentData error:", err)); // Catches user
cancellation or other errors.
  },
  // Callback for when payment is successfully authorized by the user.
  // @param {Object} paymentData - The authorized payment data from Google, including
the token.
  // @returns {Promise<Object>} A promise resolving with the final transaction
result.
  onPaymentAuthorized(paymentData) {
    return new Promise((resolve) => {
      console.log('Payment authorized. Response from Google:', paymentData);
      const paymentToken = paymentData.paymentMethodData.tokenizationData.token;
      // --- CUSTOMER ACTION REQUIRED ---
      // 1. Send this `paymentToken` to your server. Never do this from the client
side
      // 2. On your server, use this token to make the final "charge" or
"auth/capture" call to the Payment Gateway API. NEVER do this from the client side.
```

```
      // 3. Based on the response from the Payment Gateway, resolve with the final
state.
      console.log("Action required: Send this payment token to your server for
processing:", paymentToken);
      // Fetch your server here.
      resolve({ transactionState: 'SUCCESS' });
      // Example of resolving with an error if the Payment Gateway declines the
transaction:
      /*
      resolve({
        transactionState: 'ERROR',
        error: {
          intent: 'PAYMENT_AUTHORIZATION',
          message: 'Your payment was declined by the issuer. Please try another
card.',
          reason: 'PAYMENT_DATA_INVALID',
        },
      });
      */
    });
  },
}
  // Optional callback for when payment data changes (e.g., shipping address or
options).
  // @param {Object} intermediatePaymentData
  // @returns {Promise<Object>}
  function onPaymentDataChanged(intermediatePaymentData) {
    return new Promise((resolve) => {
      console.log('Intermediate payment data changed:', intermediatePaymentData);
      // This is where you would implement dynamic updates, such as recalculating
      // shipping costs or taxes based on the user's selected address.
      // For this example, we do nothing and resolve an empty object.
      resolve({});
    });
  }
```

The googlePayHandler is designed to make Google Pay work smoothly and securely.

Default Settings: It has some basic settings like the currency (USD), country (US), and it is in "TEST" mode right now (it would be "PRODUCTION" for live transactions).

Getting Merchant Information: The system must get important setup information from the store's hidden computer server. Never put sensitive payment details directly on the website itself. The store's server talks securely to the Gateway. If it can't get this information, Google Pay can't start.

Setting Up Google Pay: Once the googlePayHandler has the store's information, it sets up the connection to Google Pay.

Checking if You Can Pay with Google Pay: The system checks if the customers device and browser are ready to use Google Pay. If both are true, it shows the Google Pay button. If not, it will hide the button.

Showing the Google Pay Button: If everything is ready, the googlePayHandler puts the Google Pay button on the page where you have told it to appear.

Preparing Your Payment Request: When you click the Google Pay button, it creates a detailed request for your payment. See Configuring Google Pay.

Opening the Google Pay Window: This request is then sent to Google Pay, which makes the familiar payment window pop up on your screen.

Getting a Secure "Token": If you successfully complete your payment in the Google Pay window, the system receives a paymentToken. Think of this as a highly secure, encrypted code that represents your payment.

Security Step: It's super important that this paymentToken is never handled directly on your web browser or device. It's immediately sent to the store's secure backend server. This server is the only place that should process this Google Pay token to complete the payment with the payment company (like Inovio). Trying to do this on your device would be a huge security risk!

Finalizing the Payment: The store's server uses this paymentToken to make the final "charge" or "authorize" transaction with the payment gateway. Based on whether that's successful, your payment is completed.

Handling Changes (Optional): If you change your shipping address within Google Pay, the system can automatically update the shipping costs or taxes.

22.9 Payment Authorization

The onPaymentAuthorized() function is a critical part of your Google Pay setup. It's called automatically after the user has successfully gone through the Google Pay sheet and given their approval for the payment. This is your moment to take that secure payment information and send it to your payment processor.

```
function onPaymentAuthorized(paymentData) {
  return new Promise(function (resolve, reject) {
    // Write the data to console for debugging
    console.log('onPaymentAuthorized', paymentData);
     // --- CUSTOMER ACTION REQUIRED ---
      // 1. Send this `paymentToken` to your server.
      // 2. On your server, use this token to make the final "charge" or
"auth/capture" call to the Inoviogateway API. NEVER do this from the client side.
      // 3. Based on the response from the Inovio gateway, resolve with the final
state.
    const paymentAuthorizationResult = { transactionState: 'SUCCESS' };
     // Example of resolving with an error if your gateway declines the payment:
    /**
    const paymentAuthorizationResult ={
            transactionState: 'ERROR',
            error: {
              intent: 'PAYMENT_AUTHORIZATION',
              message: 'Insufficient funds',
              reason: 'PAYMENT_DATA_INVALID',
            },
          };
    */
    resolve(paymentAuthorizationResult);
  });
}
```

Receiving Payment Data (paymentData): When this function is called, Google Pay hands you the paymentData object. Inside this object is the secure Google payment token (located at paymentData.paymentMethodData.tokenizationData.token). This Google Pay token is what represents the customer's payment method without exposing their actual card details.

The Critical Backend Step: The most important part of this function, though commented out in the example, is where you'd send this Google payment token to your own secure server. Your server is then responsible for:

- Sending the Google Pay token to the Payment Gateway (e.g., Inovio) to authorize and capture the funds.
- Receiving the processing result from the Payment Gateway.

Returning a Promise: Just like onPaymentDataChanged(), the onPaymentAuthorized (paymentData) function returns a Promise. The Promise lets you perform asynchronous operations and then resolve with the final paymentAuthorizationResult when you have it.
In essence, onPaymentAuthorized() is the bridge between the customer approving the payment in Google Pay and your system charging their card through your payment processor.

22.10 Submitting authorized payment data

Here is an example of a request to authorize a payment with Google Pay:

```
https://api.inoviopay.com/payment/pmt_service.cfm?request_action=CCAUTHCAP&  li_count_
1=1&li_prod_id_1=111&li_value_1=49.95req_username=GATEWAY
USER&req_password=GATEWAY_PASS&site_id=11111&request_response_format=JSON&request_api_
version=4.14&request_currency=USD&PMT_WALLET=googlepay&PMT_WALLET_CRYPTOGRAM=xxxxxxx
```

22.10.1 Important requirements for Google Pay authorization requests

You must NOT include any of the following parameters, which will cause the Payment Gateway to reject your request:

```
PMT_NUMB
PMT_KEY
PMT_EXPIRY
TOKEN_GUID
PMT_ID
PMT_LAST4
PMT_ID_XTL
BILL_ADDR_STATE
BILL_ADDR_ZIP
BILL_ADDR_COUNTRY
BILL_ADDR_CITY
BILL_ADDR
```

Other gateway parameters can be included, but MUST match the values used in the Google Pay session.

- li_value_1 must match the value of paymentRequest.total in the merchantConfig data.
- request_currency must match the value of paymentRequest.currencyCode in the merchantConfig data

22.11 Rebill and Credentials on File

The Inovio response will return a CUST_ID and PMT_ID on a successful authorization using a Google Pay token (in the same way that it does for normal transactions.
To authorize on the generated Google Pay payment record, the same CUST_ID and PMT_ID should be passed in on all subsequent rebills or unscheduled merchant initiated card-on-file transaction requests.

23 Scheme Level Tokenization

Scheme tokenization is a service provided by card networks (schemes) where a payment account number (PAN) is replaced by a token. The generated network token is stored by the gateway and used for subsequent or recurring transactions. Adding scheme level tokenization service does not require additional system changes for the client.

Merchant accounts can be enrolled for scheme tokenization. For enrollment information, please contact your gateway support representative. Once a merchant account is enrolled, the gateway will automatically provision and store the network tokens for clients' customers' cards for that merchant account. Any time the card account is changed (new number, new expiry), the network token will be automatically updated. There is no action needed in the gateway API request to cause this to happen and there are no additional integration changes necessary.

Note that the gateway will relate a scheme token to a card. So if an API transaction request references a card with PMT_ID, the network token will be automatically used.

The gateway response will indicate if a card has been tokenized with the PMT_WALLET parameter. It will be set to "NETWORK TOKEN" if the card was tokenized. *Note that just because a card is tokenized does not always mean that a transaction will use the scheme token, so it is necessary to look at TRANS_NTOKEN_USED.*

The gateway response will indicate if a network token was used instead of a card for a transaction with the TRANS_NTOKEN_USED parameter.

- 0 = Scheme token was not created or used for the transaction
- 1 = Scheme token was created and used for the transaction
- 2 = Scheme token was created, but the transaction failed using the network token. Transaction was then run using the card number.

23.1 Additional Response Parameter

Field Name	Description
TRANS_NTOKEN_USED	Used to indicate whether a PAN was used instead of a Network Token when the Token fails

24 Service Provider Tokens: Merchant Decryption

24.1 Overview

This section discusses how to send credit card transactions to the Payment Gateway using service provider party wallet tokens that has been decrypted by the client. The Inovio Gateway platform supports tokens from 3rd party service providers in accordance with the EMVCo specifications. The integrated application is responsible for decoding the data-string produced by the digital wallet / e-wallet output and sending the appropriate data with the transaction request.

24.2 Request Parameters

Below is a list of transaction request parameters that are needed to route third party tokens properly.

Field Name	Description	Additional Notes
WALLET_TOKEN	Field to tell Inovio that the following transaction is using a service provider wallet provider.	Field is required. Acceptable values: ApplePay, Google Pay, or NetworkToken
WALLET_TAVV	Contains the cryptogram returned by the token provider	Field is required.
WALLET_ECI	Contains the ECI response value returned by the Token Provider.	Variable Type: Numeric Max Length: 2
WALLET_TID	Contains the token provider's token transaction identifier (TID).	Required to be sent with the transaction if returned by the Token Provider

Here is an example of a request to authorize payment with Apple Pay:

```
/payment/pmt_service.cfm?request_action=CCAUTHCAP&merch_acct_id=120603&site_id=100103&li_prod_id_1=111205&pmt_numb=************1600%20&pmt_expiry=122029&pmt_key=123&li_value_1=1.00&request_currency=USD&cust_fname=Gilberto&cust_lname=Gilberto&cust_email=test%40test.com&bill_addr_country=US&bill_addr_state=CA&bill_addr_city=Los%20Angeles&bill_addr=4022%20Randolph%20Street&bill_addr_zip=90401&XTL_IP=11.11.11.11&request_api_version=4.14&request_response_format=JSON&req_username=ellenscoffee%40gmail.com&req_password=Password123456789&argdebug=1&WALLET_ECI=07&WALLET_TID=tyujvcyug54321jksdfjk&WALLET_TAVV=%2F1QRHvYAB+88c3MOKPZ3MAACAAA%3D&WALLET_TOKEN=applepay
```

Here is an example of a request to authorize payment with Network Tokens:

```
https://api.inoviopay.com/payment/pmt_service.cfm?request_action=CCAUTHCAP&merch_acct_id=14&site_id=103&li_prod_id_1=111205&pmt_numb=45411111600&pmt_expiry=122022&pmt_key=045&li_value_1=0.01&request_currency=USD&cust_fname=Test&cust_lname=Test&cust_email=test%40test.com&bill_addr_country=US&bill_addr_state=CA&bill_addr_city=Los%20Angeles&bill_addr=4022%20Randolph%20Street&bill_addr_zip=90401&XTL_IP=11.11.11.11&request_api_version=4.14&request_response_format=JSON&req_username=test%40test.com&req_password=Password&request_initiator=C&request_rebill=2&WALLET_ECI=07&WALLET_TID=703002541600&WALLET_TAVV=AwAAAAArhfLAAAA%3D&WALLET_TOKEN=NetworkToken
```

Product functions only for applicable processors. For more information on technical usage, please contact your gateway support representative.

25 Single Euro Payments Area (SEPA)

The Payment Service supports transactions processed through Single Euro Payments Area (SEPA) Direct Debit.

Single Euro Payments Area (SEPA) Direct Debit allows Europeans with a bank account to do cashless payments online. It also provides a standardized process for payment transactions in the Euro area. For further reading and information, you may visit SEPA's website.

25.1 SEPA Direct Debit Payment Process

The following describes the lifecycle of SEPA Direct Debit transactions:

1. Customer initiates a purchase on Merchant's payment page
2. Merchant's site makes a Prepare mandate request (DBTDEBIT {"pay now" sent to trustpay}) with purchase order reference and mandate details to the gateway.
3. Gateway Response with PROC_REDIRECT_URL, transaction status is PENDING
4. Customer is redirected to PROC_REDIRECT_URL and fills up the SEPA Direct Debit mandate form in the iframe loaded on the merchant's site and confirms the data.
5. Customer is redirected to either proc_success_url or proc_error_url that were passed into the DBTDEBIT request.
6. The gateway receives a callback on the result and does a postback to merchant with the status change.

25.2 Transaction state updates through Postback

To keep merchants updated on the status of their "PENDING" SEPA transactions, the gateway will send real-time Postback to the merchant through the provided notification url.

25.3 Authorization Request (SEPA Direct Debit) - DBTAUTHORIZE

When a Mandate has been signed, A *Debit* request is used to make an authorization of payment in the customer's bank account. Merchants must send service request action, "DBTAUTHORIZE" for this type of request. 'Pay Now' is not an option on Authorization Request.

25.4 Authorization Capture Request (SEPA Direct Debit) - DBTCAPTURE

When a Mandate has been approved by the customer's bank, a capture request is made to capture the DBTAUTHORIZE 'Pay Now' for a one off sale. Merchants must send service request action, "DBTCAPTURE" for this type of request

25.5 Debit Request Parameters (DBTAUTHORIZE)

The table below describes the parameters needed to complete a **DBTAUTHORIZE** transaction request. Merchants can always send additional parameters listed in Section 4 of this specification as needed.

Field Name	Description	Data Type	Requirement
request_action	DBTAUTHORIZE	Service Request Types	Required
req_username	Service Request Username	Alphanumeric	Required
req_password	Service Request Password	Alphanumeric and Special Characters	Required
request_response_format	Service Response Format	Accepted values: "XML", "PIPES" and "JSON"	Optional (Default: XML)
request_api_version	API Version: version 4.8 and above supports European Direct Debit	Numeric	Required
site_id	Merchant's Website ID	Numeric	Required
cust_fname	Cardholder's First Name	Alphanumeric and Special Characters	Required
cust_lname	Cardholder's Last Name	Alphanumeric and Special Characters	Required
cust_email	Cardholder's Email Address	Alphanumeric	Required
li_count_1	Line Item Count Max value is "99".	Numeric	Required
li_prod_id_1	Line Item Product ID 1	Numeric	Required

li_value_1	Line Item Transaction Amount 1	Numeric	Required
bill_addr	Customer Billing Street Address	Alphanumeric	Required
bill_addr_city	Customer Billing City	Alphanumeric	Required
bill_addr_state	Customer Billing State	2-letter State or Territory Code	Some processors may require a valid 2-letter state/territory code
bill_addr_zip	Customer Billing Postal/ZIP code	Alphanumeric	Required
bill_addr_country	Customer Billing Country	2-letter Country Code ISO 3166-1 alpha-2	Required
cust_login	Customer Login or User Name	Alphanumeric and Special Characters	Optional
cust_password	Customer Password Password must be at least 10 characters with 1 number, lower case and upper case letter.	Alphanumeric	Optional
merch_acct_id	Merchant Account ID	Numeric	If null, the system will follow merchant's bank load balancer settings.
request_currency	3-letter Currency Code	All SEPA transactions are in "EUR"	Required
pmt_numb	Bank Account Number, IBAN	Alphanumeric	Required

25.5.1 Debit Request Example

```
/pmt_service.cfm?request_action=DBTAUTHORIZE&merch_acct_id=142376&site_id=116215&li_prod_id_1=128917&pmt_numb=500105175422315031&li_value_1=0.01&request_currency=EUR&cust_fname=Melinda&cust_lname=Melinda&cust_email=melinda.bozassvai%40gmail.com&bill_addr_country=DE&bill_addr_state=TW&bill_addr_city=Trierweiler&bill_addr=Unter%20den%20Buchen%201&bill_addr_zip=54311&request_api_version=4.14&request_response_format=JSON&req_username=apiuser%40example.com&req_password=Pa33w@rd&DEBIT_TYPE=SEPA'
```

25.5.2 Debit Response Example (Pending)


```
{
  "REQUEST_ACTION": "DBTAUTHORIZE",
  "REQ_ID": "18785674",
  "TRANS_STATUS_NAME": "PENDING",
  "TRANS_VALUE": 0.01,
```

```
  "CURR_CODE_ALPHA": "EUR",
  "TRANS_VALUE_SETTLED": 0.01,
  "CURR_CODE_ALPHA_SETTLED": "EUR",
  "TRANS_EXCH_RATE": "",
  "TRANS_ID": 2001878902,
  "CUST_ID": 7534577,
  "XTL_CUST_ID": "",
  "PO_ID": 18103630,
  "XTL_ORDER_ID": "",
  "BATCH_ID": 140418,
  "PROC_NAME": "TrustPay",
  "MERCH_ACCT_ID": 142376,
  "CARD_BRAND_NAME": "SEPA",
  "CARD_TYPE": "",
  "CARD_PREPAID": "",
  "CARD_BANK": "",
  "CARD_DETAIL": "",
  "CARD_BALANCE": "",
  "PMT_L4": "5031",
  "PMT_ID": 4354240,
  "PMT_ID_XTL": "",
  "PMT_AAU_UPDATE_DT": "",
  "PMT_AAU_UPDATE_DESC": "",
  "PROC_UDF01": "",
  "PROC_UDF02": "",
  "PROC_AUTH_RESPONSE": "",
  "PROC_RETRIEVAL_NUM": "",
  "PROC_REFERENCE_NUM": "",
  "PROC_REDIRECT_URL": "https://aapi.trustpay.eu/SepaDirectDebitMandates/input/659b33f4-f5d8-4acb-9dc7-6673e43e57ae",
  "AVS_RESPONSE": "",
  "CVV_RESPONSE": "",
  "CARD_BRAND_TRANSID": "",
  "REQUEST_API_VERSION": "4.14",
  "P3DS_VENDOR": "",
  "P3DS_RESPONSE": "",
  "PO_LI_ID_1": "1901856",
  "PO_LI_COUNT_1": 1,
  "PO_LI_AMOUNT_1": "0.01",
  "PO_LI_PROD_ID_1": "128917",
  "MBSHP_ID_1": "",
  "TRANS_NTOKEN_USED": 0
}
```


25.5.3 Debit Response Example (Success)


```
{
  "REQUEST_ACTION": "DBTCAPTURE",
  "REQ_ID": "18787081",
  "TRANS_STATUS_NAME": "APPROVED",
  "TRANS_VALUE": 0.01,
  "CURR_CODE_ALPHA": "EUR",
  "TRANS_VALUE_SETTLED": 0.01,
  "CURR_CODE_ALPHA_SETTLED": "EUR",
```

```
  "TRANS_EXCH_RATE": "",
  "TRANS_ID": 2001878992,
  "CUST_ID": 7534577,
  "XTL_CUST_ID": "",
  "PO_ID": 18103673,
  "XTL_ORDER_ID": "",
  "BATCH_ID": 140418,
  "PROC_NAME": "TrustPay",
  "MERCH_ACCT_ID": 142376,
  "CARD_BRAND_NAME": "SEPA",
  "CARD_TYPE": "",
  "CARD_PREPAID": "",
  "CARD_BANK": "",
  "CARD_DETAIL": "",
  "CARD_BALANCE": "",
  "PMT_L4": "5031",
  "PMT_ID": 4354240,
  "PMT_ID_XTL": "",
  "PMT_AAU_UPDATE_DT": "",
  "PMT_AAU_UPDATE_DESC": "",
  "PROC_UDF01": "",
  "PROC_UDF02": "",
  "PROC_AUTH_RESPONSE": "",
  "PROC_RETRIEVAL_NUM": "6214013397",
  "PROC_REFERENCE_NUM": "",
  "PROC_REDIRECT_URL": "",
  "AVS_RESPONSE": "",
  "CVV_RESPONSE": "",
  "CARD_BRAND_TRANSID": "",
  "REQUEST_API_VERSION": "4.14",
  "P3DS_VENDOR": "",
  "P3DS_RESPONSE": "",
  "PO_LI_ID_1": "1901896",
  "PO_LI_COUNT_1": 1,
  "PO_LI_AMOUNT_1": "0.01",
  "PO_LI_PROD_ID_1": "128917",
  "MBSHP_ID_1": "",
  "TRANS_NTOKEN_USED": 0
}
```


25.6 Reversals

25.6.1 Reversing a mandate (DBTREVERSE)

Merchants or customers can choose to cancel existing mandate whenever they want. To cancel an existing Mandate, Merchants should use DBTREVERSE Request_Action. This request_action also stops future rebills and cancels subscriptions.

25.6.2 DBTREVERSE transaction Request Example


```
/payment/pmt_service.cfm?request_action=DBTREVERSE&merch_acct_id=123456&site_id=456878&li_prod_id_1=14512&pmt_numb=200125175423005031&li_value_1=0.01&request_currency=EUR&cust_fname=CustomerFN&cust_lname=Masha&cust_email=msepauser%40gmail.com&bill_addr_country=DE&bill_addr_state=TW&bill_addr_city=Trierweiler&bill_addr=Unter%20den%20mystreetn%201&bill_addr_zip=54311&XT
```

```
L_IP=11.11.11.11&request_api_version=4.14&request_response_format=JSON&req_username=puser%40exa
mple.com&req_password=Testingpassword&argdebug=1&PROC_SUCCESS_URL=testingSuccess&PROC_E
RROR_URL=testingError&request_ref_po_id=18103694'
```

25.7 Refunds or Credits

25.7.1 Issuing refunds or credits (DBTCREDIT)

Merchants may request to credit a direct debit order by sending **DBTCREDIT** in the request_action parameter. Note that the request may not be processed successfully by the bank for certain transaction states (e.g. you cannot credit a payment that has not been collected by the bank).

To do a DBTCREDIT request, send the Order ID (PO_ID) of the original authorization in the **REQUEST_REF_PO_ID** parameter.

DBTCREDIT Required Fields

SITE_ID
REQ_USERNAME
REQ_PASSWORD
REQUEST_ACTION = DBTCREDIT
REQUEST_RESPONSE_FORMAT
REQUEST_REF_PO_ID
REQUEST_API_VERSION

25.8 Recurring transactions

SEPA Direct Debit supports recurring payments. Recurring payment Mandate Type authorizes subsequent rebills until canceled. Request Action **DBTDEBIT** is used for initial and subsequent rebills as authorized by Mandate. In summary these are the Recurring transactions scenarios.

- DBTDEBIT & request_rebill=2: start of rebill subscription
- DBTDEBIT & request_rebill=1 & cust_id & pmt_id: rebill payment #1
- DBTDEBIT & request_rebill=1 & cust_id & pmt_id: rebill payment #2
- DBTREVERSE: cancel mandate, end of subscription

25.8.1 Recurring transaction Request Example (DBTDEBIT & request_rebill=2)

```
/payment/pmt_service.cfm?request_action=DBTDEBIT&merch_acct_id=142376&site_id=116215&li_prod_id
_1=128917&pmt_numb=300105172002111031&li_value_1=0.01&request_currency=EUR&cust_fname=Masha
&cust_lname=cust&cust_email=customer.sepa%40example1.com&bill_addr_country=DE&bill_addr_state=
TW&bill_addr_city=Trierweiler&bill_addr=Unter%20den%20mystreetn%201&bill_addr_zip=54311&XTL_IP
=11.11.11.11&request_api_version=4.14&request_response_format=JSON&req_username=puser%40example.
com&req_password=Oosii499&request_ref_po_id=18103687&request_rebill=2'
```

25.8.2 Recurring transaction Response Example (DBTDEBIT & request_rebill=2)


```
"REQUEST_ACTION": "DBTDEBIT",
"REQ_ID": "18785674",
"TRANS_STATUS_NAME": "PENDING",
"TRANS_VALUE": 0.01,
"CURR_CODE_ALPHA": "EUR",
```

```
  "TRANS_VALUE_SETTLED": 0.01,
  "CURR_CODE_ALPHA_SETTLED": "EUR",
  "TRANS_EXCH_RATE": "",
  "TRANS_ID": 2001878902,
  "CUST_ID": 7534577,
  "XTL_CUST_ID": "",
  "PO_ID": 18103630,
  "XTL_ORDER_ID": "",
  "BATCH_ID": 140418,
  "PROC_NAME": "TrustPay",
  "MERCH_ACCT_ID": 142376,
  "CARD_BRAND_NAME": "EUDD",
  "CARD_TYPE": "",
  "CARD_PREPAID": "",
  "CARD_BANK": "",
  "CARD_DETAIL": "",
  "CARD_BALANCE": "",
  "PMT_L4": "5031",
  "PMT_ID": 4354240,
  "PMT_ID_XTL": "",
  "PMT_AAU_UPDATE_DT": "",
  "PMT_AAU_UPDATE_DESC": "",
  "PROC_UDF01": "",
  "PROC_UDF02": "",
  "PROC_AUTH_RESPONSE": "",
  "PROC_RETRIEVAL_NUM": "",
  "PROC_REFERENCE_NUM": "",
  "PROC_REDIRECT_URL": "https://aapi.trustpay.eu/SepaDirectDebitMandates/input/659b33f4-f5d8-4acb-
9dc7-6673e43e57ae",
  "AVS_RESPONSE": "",
  "CVV_RESPONSE": "",
  "CARD_BRAND_TRANSID": "",
  "REQUEST_API_VERSION": "4.14",
  "P3DS_VENDOR": "",
  "P3DS_RESPONSE": "",
  "PO_LI_ID_1": "1901856",
  "PO_LI_COUNT_1": 1,
  "PO_LI_AMOUNT_1": "0.01",
  "PO_LI_PROD_ID_1": "128917",
  "MBSHP_ID_1": "",
  "TRANS_NTOKEN_USED": 0
}
```


25.8.3 Recurring transaction Request Example (DBTDEBIT & request_rebill=1)


```
/payment/pmt_service.cfm?request_action=DBTDEBIT&merch_acct_id=142376&site_id=116215&li_prod_id
_1=128917&pmt_numb=200100075433315031&li_value_1=0.01&request_currency=EUR&cust_fname=Masha
da&cust_lname=Sam&cust_email=customer%40gmail.com&bill_addr_country=DE&bill_addr_state=TW&b
ill_addr_city=Trierweiler&bill_addr=Unter%20den%20mystreet%201&bill_addr_zip=54311&request_api_ve
rsion=4.14&request_response_format=JSON&req_username=apiuser%40example.com&req_password=Pas
swormed@98&request_rebill=1&CUST_ID=7534577&PMT_ID=4354240'
```

CUST_ID and PMT_ID are passed into the Gateway. Gateway finds the original mandate, and charges the customer

Recurring Transaction Fields

SITE_ID
REQ_USERNAME
REQ_PASSWORD
REQUEST_ACTION
REQUEST_RESPONSE_FORMAT
REQUEST_API_VERSION
CUST_ID
LI_PROD_ID_1
LI_VALUE_1
LI_COUNT_1
PMT_ID
MERCH_ACCT_ID
REQUEST_CURRENCY

25.8.4 Recurring transaction Response (DBTDEBIT & request_rebill=1) Example


```
"REQUEST_ACTION": "DBTDEBIT",
"REQ_ID": "18787586",
"TRANS_STATUS_NAME": "APPROVED",
"TRANS_VALUE": 0.01,
"CURR_CODE_ALPHA": "EUR",
"TRANS_VALUE_SETTLED": 0.01,
"CURR_CODE_ALPHA_SETTLED": "EUR",
"TRANS_EXCH_RATE": "",
"TRANS_ID": 2001879023,
"CUST_ID": 7534577,
"XTL_CUST_ID": "",
"PO_ID": 18103688,
"XTL_ORDER_ID": "",
"BATCH_ID": 140418,
"PROC_NAME": "TrustPay",
"MERCH_ACCT_ID": 142376,
"CARD_BRAND_NAME": "EUDD",
"CARD_TYPE": "",
"CARD_PREPAID": "",
"CARD_BANK": "",
"CARD_DETAIL": "",
"CARD_BALANCE": "",
"PMT_L4": "5031",
"PMT_ID": 4354240,
"PMT_ID_XTL": "",
"PMT_AAU_UPDATE_DT": "",
"PMT_AAU_UPDATE_DESC": "",
"PROC_UDF01": "",
"PROC_UDF02": "",
"PROC_AUTH_RESPONSE": "",
"PROC_RETRIEVAL_NUM": "5602944007",
"PROC_REFERENCE_NUM": "",
"PROC_REDIRECT_URL": "",
"AVS_RESPONSE": "",
```

```
  "CVV_RESPONSE": "",
  "CARD_BRAND_TRANSID": "",
  "REQUEST_API_VERSION": "4.14",
  "P3DS_VENDOR": "",
  "P3DS_RESPONSE": "",
  "PO_LI_ID_1": "1901910",
  "PO_LI_COUNT_1": 1,
  "PO_LI_AMOUNT_1": "0.01",
  "PO_LI_PROD_ID_1": "128917",
  "MBSHP_ID_1": "",
  "TRANS_NTOKEN_USED": 0
}
```


25.9 Additional Parameters

Field Name	Description
PROC_SUCCESS_URL	Used to indicate the URL to which the customer is redirected after successful payment (if PayNow was chosen)
PROC_ERROR_URL	Used to indicate the URL to which the customer is redirected after an error or failed payment (if PayNow was chosen)
PMT_NUMB	Used for IBAN Field (Required)
DEBIT_TYPE	Used to indicate the Debit type used for payment (SEPA, iDEAL, EPS) - Required

26 iDEAL Payments

iDEAL is a Dutch (Netherlands-NL) -based payment method that allows customers to complete transactions online using their bank credentials. As such, the customer's bank account has to be in the Netherlands for iDEAL payment availability. When a customer selects to pay with iDEAL, they're presented with a list of banks that support iDEAL. After the customer selects their bank, they're redirected to the bank's website or mobile app to complete the payment. iDEAL is the most commonly used online banking method in the Netherlands, supported by all the major Dutch consumer banks.

26.1 Debit Request Parameters (DBTAUTHORIZE)

The table below describes the parameters needed to complete a **DBTAUTHORIZE** transaction request. Merchants can always send additional parameters listed in Section 4 of our API Documentation as needed.

Field Name	Description	Data Type	Requirement
request_action	DBTAUTHORIZE	Service Request Types	Required
req_username	Service Request Username	Alphanumeric	Required

req_password	Service Request Password	Alphanumeric and Special Characters	Required
request_response_format	Service Response Format	Accepted values: “XML”, “PIPES” and ”JSON”	Optional (Default: XML)
request_api_version	API Version: version 4.8 and above supports European Direct Debit	Numeric	Required
site_id	Merchant’s Website ID	Numeric	Required
cust_fname	Cardholder’s First Name	Alphanumeric and Special Characters	Required
cust_lname	Cardholder’s Last Name	Alphanumeric and Special Characters	Required
cust_email	Cardholder’s Email Address	Alphanumeric	Required
li_count_1	Line Item Count Max value is “99”.	Numeric	Required
li_prod_id_1	Line Item Product ID 1	Numeric	Required
li_value_1	Line Item Transaction Amount 1	Numeric	Required
bill_addr	Customer Billing Street Address	Alphanumeric	Required
bill_addr_city	Customer Billing City	Alphanumeric	Required
bill_addr_state	Customer Billing State	2-letter State or Territory Code	Some processors may require a valid 2-letter state/territory code
bill_addr_zip	Customer Billing Postal/ZIP code	Alphanumeric	Required
bill_addr_country	Customer Billing Country	2-letter Country Code ISO 3166-1 alpha-2	Required

cust_login	Customer Login or User Name	Alphanumeric and Special Characters	Optional
cust_password	Customer Password Password must be at least 10 characters with 1 number, lower case and upper case letter.	Alphanumeric	Optional
merch_acct_id	Merchant Account ID	Numeric	If null, the system will follow merchant's bank load balancer settings.
request_currency	3-letter Currency Code	All iDEAL transactions are in "EUR"	Required
pmt_numb	Bank Account Number, IBAN	Alphanumeric	Required

26.1.1 Debit Request Example

```
https://api.inoviopay.com/payment/pmt_service.cfm
?request_action=DBTAUTHORIZE&merch_acct_id=142376&site_id=116215&li_prod_id_1=128917&pmt_nu
mb=5001051754223150 31&li_value_1=0.01&request_currency=EUR&cust_fname=Melinda&cust_lname=Mel
inda&cust_email=melinda.bozassvai%40gmail.com&bill_addr_country=NL&bill_addr_state=TW&bill_addr
_city=NLCITY&bill_addr=Unter%20den%20Buchen%201&bill_addr_zip=54311&request_api_version=4.14&r
equest_response_format=JSON&req_username=apiuser%40example.com&req_password=Pa33w@rd&DEBI
T_TYPE=iDEAL'
```

26.1.2 Debit Response Example (Pending)


```
{
  "REQUEST_ACTION": "DBTAUTHORIZE",
  "REQ_ID": "18780074",
  "TRANS_STATUS_NAME": "PENDING",
  "TRANS_VALUE": 0.01,
  "CURR_CODE_ALPHA": "EUR",
  "TRANS_VALUE_SETTLED": 0.01,
  "CURR_CODE_ALPHA_SETTLED": "EUR",
  "TRANS_EXCH_RATE": "",
  "TRANS_ID": 2001878902,
  "CUST_ID": 7534577,
  "XTL_CUST_ID": "",
  "PO_ID": 18103630,
  "XTL_ORDER_ID": "",
  "BATCH_ID": 140418,
  "PROC_NAME": "TrustPay",
  "MERCH_ACCT_ID": 142376,
  "DEBIT_TYPE": "iDEAL",
  "CARD_TYPE": "",
  "CARD_PREPAID": "",
```

```
  "CARD_BANK": "",
  "CARD_DETAIL": "",
  "CARD_BALANCE": "",
  "PMT_L4": "5031",
  "PMT_ID": 4354240,
  "PMT_ID_XTL": "",
  "PMT_AAU_UPDATE_DT": "",
  "PMT_AAU_UPDATE_DESC": "",
  "PROC_UDF01": "",
  "PROC_UDF02": "",
  "PROC_AUTH_RESPONSE": "",
  "PROC_RETRIEVAL_NUM": "",
  "PROC_REFERENCE_NUM": "",
  "PROC_REDIRECT_URL": "https://aapi.trustpay.eu/SepaDirectDebitMandates/input/659b33f4-f5d8-4acb-9dc7-6673e43e57ae",
  "AVS_RESPONSE": "",
  "CVV_RESPONSE": "",
  "CARD_BRAND_TRANSID": "",
  "REQUEST_API_VERSION": "4.14",
  "P3DS_VENDOR": "",
  "P3DS_RESPONSE": "",
  "PO_LI_ID_1": "1901856",
  "PO_LI_COUNT_1": 1,
  "PO_LI_AMOUNT_1": "0.01",
  "PO_LI_PROD_ID_1": "128917",
  "MBSHP_ID_1": "",
  "TRANS_NTOKEN_USED": 0
}
```


26.1.1 Debit Response Example (Success)


```
  "REQUEST_ACTION": "DBTCAPTURE",
  "REQ_ID": "18787081",
  "TRANS_STATUS_NAME": "APPROVED",
  "TRANS_VALUE": 0.01,
  "CURR_CODE_ALPHA": "EUR",
  "TRANS_VALUE_SETTLED": 0.01,
  "CURR_CODE_ALPHA_SETTLED": "EUR",
  "TRANS_EXCH_RATE": "",
  "TRANS_ID": 2001878992,
  "CUST_ID": 7534577,
  "XTL_CUST_ID": "",
  "PO_ID": 18103673,
  "XTL_ORDER_ID": "",
  "BATCH_ID": 140418,
  "PROC_NAME": "TrustPay",
  "MERCH_ACCT_ID": 142376,
  "DEBIT_TYPE": "iDEAL",
  "CARD_TYPE": "",
  "CARD_PREPAID": "",
  "CARD_BANK": "",
  "CARD_DETAIL": "",
  "CARD_BALANCE": "",
```

```
  "PMT_L4": "5031",
  "PMT_ID": 4354240,
  "PMT_ID_XTL": "",
  "PMT_AAU_UPDATE_DT": "",
  "PMT_AAU_UPDATE_DESC": "",
  "PROC_UDF01": "",
  "PROC_UDF02": "",
  "PROC_AUTH_RESPONSE": "",
  "PROC_RETRIEVAL_NUM": "6214013397",
  "PROC_REFERENCE_NUM": "",
  "PROC_REDIRECT_URL": "",
  "AVS_RESPONSE": "",
  "CVV_RESPONSE": "",
  "CARD_BRAND_TRANSID": "",
  "REQUEST_API_VERSION": "4.14",
  "P3DS_VENDOR": "",
  "P3DS_RESPONSE": "",
  "PO_LI_ID_1": "1901896",
  "PO_LI_COUNT_1": 1,
  "PO_LI_AMOUNT_1": "0.01",
  "PO_LI_PROD_ID_1": "128917",
  "MBSHP_ID_1": "",
  "TRANS_NTOKEN_USED": 0
}
```


26.2 Reversals

26.2.1 Reversing a mandate (DBTREVERSE)

Merchants or customers can choose to cancel existing mandate whenever they want. To cancel an existing Mandate, Merchants should use DBTREVERSE Request_Action. This request_action also stops future rebills and cancels subscriptions.

26.2.2 DBTREVERSE transaction Request Example

```
https://api.inoviopay.com/payment/pmt_service.cfm
?request_action=DBTREVERSE&merch_acct_id=123456&site_id=456878&li_prod_id_1=14512&pmt_numb=
400125175423005031&li_value_1=0.01&request_currency=EUR&cust_fname=CustomerFN&cust_lname=Ma
sha&cust_email=msepauser%40gmail.com&bill_addr_country=NL&bill_addr_state=TW&bill_addr_city=NL
CITY&bill_addr=Unter%20den%20mystreetn%201&bill_addr_zip=54311&XTL_IP=11.11.11.11&request_api_ve
rsion=4.14&request_response_format=JSON&req_username=puser%40example.com&req_password=Testi
ngpassword&argdebug=1&PROC_SUCCESS_URL=testingSuccess&PROC_ERROR_URL=testingError&requ
est_ref_po_id=18103694'
```

26.3 Refunds or Credits

26.3.1 Issuing refunds or credits (DBTCREDIT)

Merchants may request to credit a direct debit order by sending **DBTCREDIT** in the request_action parameter. Note that the request may not be processed successfully by the bank for certain transaction states (e.g. you cannot credit a payment that has not been collected by the bank).

To do a DBTCREDIT request, send the Order ID (PO_ID) of the original authorization in the **REQUEST_REF_PO_ID** parameter.

DBTCREDIT Required Fields

SITE_ID
REQ_USERNAME
REQ_PASSWORD
REQUEST_ACTION = DBTCREDIT
REQUEST_RESPONSE_FORMAT
REQUEST_REF_PO_ID
REQUEST_API_VERSION

26.4 Recurring transactions

iDEAL supports recurring payments through SEPA. Recurring payment Mandate Type authorizes subsequent rebills until canceled. Request Action **DBTDEBIT** is used for initial and subsequent rebills as authorized by Mandate. In summary these are the Recurring transactions scenarios.

- DBTDEBIT & request_rebill=2: start of rebill subscription
- DBTDEBIT & request_rebill=1 & cust_id & pmt_id: rebill payment #1
- DBTDEBIT & request_rebill=1 & cust_id & pmt_id: rebill payment #2
- DBTREVERSE: cancel mandate, end of subscription

26.4.1 Recurring transaction Request Example (DBTDEBIT & request_rebill=2)

```
https://api.inoviopay.com/payment/pmt_service.cfm
?request_action=DBTDEBIT&merch_acct_id=142376&site_id=116215&li_prod_id_1=128917&pmt_numb=30
01051720021110 31&li_value_1=0.01&request_currency=EUR&cust_fname=Masha&cust_lname=cust&cust_e
mail=customer.sepa%40example1.com&bill_addr_country=NL&bill_addr_state=TW&bill_addr_city=NLCIT
Y&bill_addr=Unter%20den%20mystreetn%201&bill_addr_zip=54311&XTL_IP=11.11.11.11&request_api_versio
n=4.14&request_response_format=JSON&req_username=puser%40example.com&req_password=Oosii499
&request_ref_po_id=18103687&request_rebill=2'
```

26.4.2 Recurring transaction Response Example (DBTDEBIT & request_rebill=2)

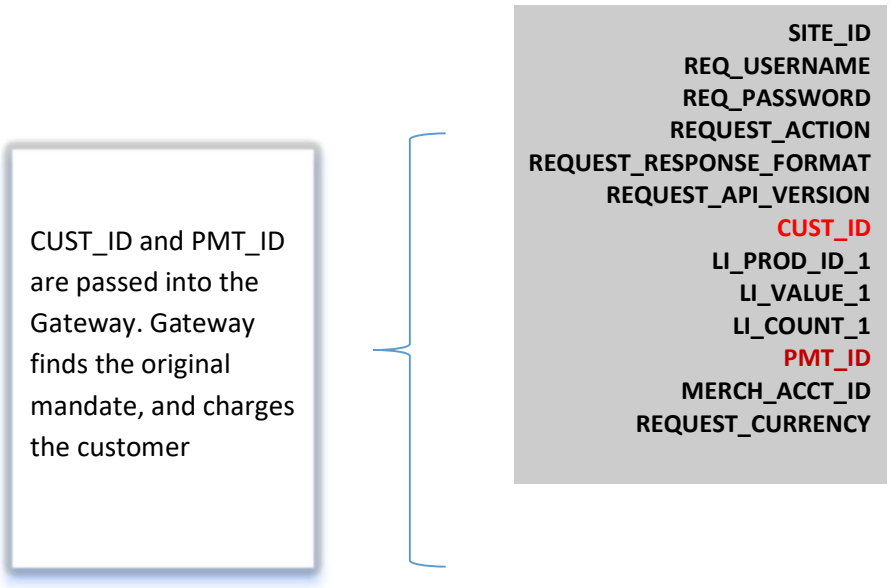

```
"REQUEST_ACTION": "DBTDEBIT",
"REQ_ID": "18785674",
"TRANS_STATUS_NAME": "PENDING",
"TRANS_VALUE": 0.01,
"CURR_CODE_ALPHA": "EUR",
"TRANS_VALUE_SETTLED": 0.01,
"CURR_CODE_ALPHA_SETTLED": "EUR",
"TRANS_EXCH_RATE": "",
"TRANS_ID": 2001878902,
"CUST_ID": 7534577,
"XTL_CUST_ID": "",
"PO_ID": 18103630,
"XTL_ORDER_ID": "",
"BATCH_ID": 140418,
"PROC_NAME": "TrustPay",
"MERCH_ACCT_ID": 142376,
"DEBIT_TYPE": "iDEAL",
```

```
  "CARD_TYPE": "",
  "CARD_PREPAID": "",
  "CARD_BANK": "",
  "CARD_DETAIL": "",
  "CARD_BALANCE": "",
  "PMT_L4": "5031",
  "PMT_ID": 4354240,
  "PMT_ID_XTL": "",
  "PMT_AAU_UPDATE_DT": "",
  "PMT_AAU_UPDATE_DESC": "",
  "PROC_UDF01": "",
  "PROC_UDF02": "",
  "PROC_AUTH_RESPONSE": "",
  "PROC_RETRIEVAL_NUM": "",
  "PROC_REFERENCE_NUM": "",
  "PROC_REDIRECT_URL": "https://aapi.trustpay.eu/SepaDirectDebitMandates/input/659b33f4-f5d8-4acb-9dc7-6673e43e57ae",
  "AVS_RESPONSE": "",
  "CVV_RESPONSE": "",
  "CARD_BRAND_TRANSID": "",
  "REQUEST_API_VERSION": "4.14",
  "P3DS_VENDOR": "",
  "P3DS_RESPONSE": "",
  "PO_LI_ID_1": "1901856",
  "PO_LI_COUNT_1": 1,
  "PO_LI_AMOUNT_1": "0.01",
  "PO_LI_PROD_ID_1": "128917",
  "MBSHP_ID_1": "",
  "TRANS_NTOKEN_USED": 0
}
```


26.4.3 Recurring transaction Request Example (DBTDEBIT & request_rebill=1)


```
https://api.inoviopay.com/payment/pmt_service.cfm
?request_action=DBTDEBIT&merch_acct_id=142376&site_id=116215&li_prod_id_1=128917&pmt_numb=400100007543331503 1&li_value_1=0.01&request_currency=EUR&cust_fname=Mashada&cust_lname=Sam&cust_email=customer%40gmail.com&bill_addr_country=NL&bill_addr_state=TW&bill_addr_city=NLCITY&bill_addr=Unter%20den%20mystreet%201&bill_addr_zip=54311&request_api_version=4.14&request_response_format=JSON&req_username=apiuser%40example.com&req_password=Passwormed@98&request_rebill=1&CUST_ID=7534577&PMT_ID=4354240'
```


Recurring Transaction Fields



26.4.4 Recurring transaction Response Example (DBTDEBIT & request_rebill=1)


```
"REQUEST_ACTION": "DBTDEBIT",
"REQ_ID": "18787586",
"TRANS_STATUS_NAME": "APPROVED",
"TRANS_VALUE": 0.01,
"CURR_CODE_ALPHA": "EUR",
"TRANS_VALUE_SETTLED": 0.01,
"CURR_CODE_ALPHA_SETTLED": "EUR",
"TRANS_EXCH_RATE": "",
"TRANS_ID": 2001879023,
"CUST_ID": 7534577,
"XTL_CUST_ID": "",
"PO_ID": 18103688,
"XTL_ORDER_ID": "",
"BATCH_ID": 140418,
"PROC_NAME": "TrustPay",
"MERCH_ACCT_ID": 142376,
"DEBIT_TYPE": "iDEAL",
"CARD_TYPE": "",
"CARD_PREPAID": "",
"CARD_BANK": "",
"CARD_DETAIL": "",
"CARD_BALANCE": "",
"PMT_L4": "5031",
"PMT_ID": 4354240,
"PMT_ID_XTL": "",
"PMT_AAU_UPDATE_DT": "",
"PMT_AAU_UPDATE_DESC": "",
"PROC_UDF01": "",
"PROC_UDF02": "",
"PROC_AUTH_RESPONSE": "",
"PROC_RETRIEVAL_NUM": "5602944007",
"PROC_REFERENCE_NUM": "",
```

```
  "PROC_REDIRECT_URL": "",
  "AVS_RESPONSE": "",
  "CVV_RESPONSE": "",
  "CARD_BRAND_TRANSID": "",
  "REQUEST_API_VERSION": "4.14",
  "P3DS_VENDOR": "",
  "P3DS_RESPONSE": "",
  "PO_LI_ID_1": "1901910",
  "PO_LI_COUNT_1": 1,
  "PO_LI_AMOUNT_1": "0.01",
  "PO_LI_PROD_ID_1": "128917",
  "MBSHP_ID_1": "",
  "TRANS_NTOKEN_USED": 0
}
```


26.5 Additional Parameters

Field Name	Description
PROC_SUCCESS_URL	Used to indicate the URL to which the customer is redirected after successful payment (if PayNow was chosen)
PROC_ERROR_URL	Used to indicate the URL to which the customer is redirected after an error or failed payment (if PayNow was chosen)
PMT_NUMB	Used for IBAN Field (Required)
DEBIT_TYPE	Used to indicate the Debit type used for payment (SEPA, iDEAL, EPS) - Required

27 EPS Payments

Electronic Payment Standard (EPS) is an Austrian online banking payment method operated by the Austrian Banking Association. EPS is a guaranteed payment method which lets customers make online payments through their online banking. EPS Payment checkout process works as follows;

1. Customer Bank Account must be located in Austria for EPS payment availability
2. Merchant can be located anywhere within the approved country with local currency presentment
3. Merchant offers EPS as payment method on their checkout page
4. Customer selects EPS from the list of payment methods, then selects their bank.
5. Customer is redirected to the bank website to enter their bank details.
6. Customer confirms the payment and is redirected back to checkout page.

27.1 Debit Request Parameters (DBTAUTHORIZE)

The table below describes the parameters needed to complete a **DBTAUTHORIZE** transaction request. Merchants can always send additional parameters listed in Section 4 of our API Documentation as needed.

Field Name	Description	Data Type	Requirement

request_action	DBTAUTHORIZE	Service Request Types	Required
req_username	Service Request Username	Alphanumeric	Required
req_password	Service Request Password	Alphanumeric and Special Characters	Required
request_response_format	Service Response Format	Accepted values: “XML”, “PIPES” and ”JSON”	Optional (Default: XML)
request_api_version	API Version: version 4.8 and above supports European Direct Debit	Numeric	Required
site_id	Merchant’s Website ID	Numeric	Required
cust_fname	Cardholder’s First Name	Alphanumeric and Special Characters	Required
cust_lname	Cardholder’s Last Name	Alphanumeric and Special Characters	Required
cust_email	Cardholder’s Email Address	Alphanumeric	Required
li_count_1	Line Item Count Max value is “99”.	Numeric	Required
li_prod_id_1	Line Item Product ID 1	Numeric	Required
li_value_1	Line Item Transaction Amount 1	Numeric	Required
bill_addr	Customer Billing Street Address	Alphanumeric	Required
bill_addr_city	Customer Billing City	Alphanumeric	Required
bill_addr_state	Customer Billing State	2-letter State or Territory Code	Some processors may require a valid 2-letter state/territory code

bill_addr_zip	Customer Billing Postal/ZIP code	Alphanumeric	Required
bill_addr_country	Customer Billing Country	2-letter Country Code ISO 3166-1 alpha-2	Required
cust_login	Customer Login or User Name	Alphanumeric and Special Characters	Optional
cust_password	Customer Password Password must be at least 10 characters with 1 number, lower case and upper case letter.	Alphanumeric	Optional
merch_acct_id	Merchant Account ID	Numeric	If null, the system will follow merchant's bank load balancer settings.
request_currency	3-letter Currency Code	All EPS transactions are in "EUR"	Required
pmt_numb	Bank Account Number, IBAN	Alphanumeric	Required

27.1.1 Debit Request Example

```
https://api.inoviopay.com/payment/pmt_service.cfm
?request_action=DBTAUTHORIZE&merch_acct_id=142376&site_id=116215&li_prod_id_1=128917&pmt_nu
mb=500105175422315031&li_value_1=0.01&request_currency=EUR&cust_fname=Melinda&cust_lname=Mel
inda&cust_email=melinda.bozassvai%40gmail.com&bill_addr_country=AT&bill_addr_state=TW&bill_addr
_city=Trierweiler&bill_addr=Unter%20den%20Buchen%201&bill_addr_zip=54311&request_api_version=4.1
4&request_response_format=JSON&req_username=apiuser%40example.com&req_password=Pa33w@rd&D
EBIT_TYPE=EPS'
```

27.1.2 Debit Response Example (Pending)


```
{
  "REQUEST_ACTION": "DBTAUTHORIZE",
  "REQ_ID": "18785674",
  "TRANS_STATUS_NAME": "PENDING",
  "TRANS_VALUE": 0.01,
  "CURR_CODE_ALPHA": "EUR",
  "TRANS_VALUE_SETTLED": 0.01,
  "CURR_CODE_ALPHA_SETTLED": "EUR",
  "TRANS_EXCH_RATE": "",
  "TRANS_ID": 2001878902,
  "CUST_ID": 7534577,
  "XTL_CUST_ID": "",
  "PO_ID": 18103630,
```

```
  "XTL_ORDER_ID": "",
  "BATCH_ID": 140418,
  "PROC_NAME": "TrustPay",
  "MERCH_ACCT_ID": 142376,
  "DEBIT_TYPE": "EPS",
  "CARD_TYPE": "",
  "CARD_PREPAID": "",
  "CARD_BANK": "",
  "CARD_DETAIL": "",
  "CARD_BALANCE": "",
  "PMT_L4": "5031",
  "PMT_ID": 4354240,
  "PMT_ID_XTL": "",
  "PMT_AAU_UPDATE_DT": "",
  "PMT_AAU_UPDATE_DESC": "",
  "PROC_UDF01": "",
  "PROC_UDF02": "",
  "PROC_AUTH_RESPONSE": "",
  "PROC_RETRIEVAL_NUM": "",
  "PROC_REFERENCE_NUM": "",
  "PROC_REDIRECT_URL": "https://aapi.trustpay.eu/SepaDirectDebitMandates/input/659b33f4-f5d8-4acb-9dc7-6673e43e57ae",
  "AVS_RESPONSE": "",
  "CVV_RESPONSE": "",
  "CARD_BRAND_TRANSID": "",
  "REQUEST_API_VERSION": "4.14",
  "P3DS_VENDOR": "",
  "P3DS_RESPONSE": "",
  "PO_LI_ID_1": "1901856",
  "PO_LI_COUNT_1": 1,
  "PO_LI_AMOUNT_1": "0.01",
  "PO_LI_PROD_ID_1": "128917",
  "MBSHP_ID_1": "",
  "TRANS_NTOKEN_USED": 0
}
```


27.1.1 Debit Response Example (Success)


```
  "REQUEST_ACTION": "DBTCAPTURE",
  "REQ_ID": "18787081",
  "TRANS_STATUS_NAME": "APPROVED",
  "TRANS_VALUE": 0.01,
  "CURR_CODE_ALPHA": "EUR",
  "TRANS_VALUE_SETTLED": 0.01,
  "CURR_CODE_ALPHA_SETTLED": "EUR",
  "TRANS_EXCH_RATE": "",
  "TRANS_ID": 2001878992,
  "CUST_ID": 7534577,
  "XTL_CUST_ID": "",
  "PO_ID": 18103673,
  "XTL_ORDER_ID": "",
  "BATCH_ID": 140418,
  "PROC_NAME": "TrustPay",
```

```
  "MERCH_ACCT_ID": 142376,
  "DEBIT_TYPE": "EPS",
  "CARD_TYPE": "",
  "CARD_PREPAID": "",
  "CARD_BANK": "",
  "CARD_DETAIL": "",
  "CARD_BALANCE": "",
  "PMT_L4": "5031",
  "PMT_ID": 4354240,
  "PMT_ID_XTL": "",
  "PMT_AAU_UPDATE_DT": "",
  "PMT_AAU_UPDATE_DESC": "",
  "PROC_UDF01": "",
  "PROC_UDF02": "",
  "PROC_AUTH_RESPONSE": "",
  "PROC_RETRIEVAL_NUM": "6214013397",
  "PROC_REFERENCE_NUM": "",
  "PROC_REDIRECT_URL": "",
  "AVS_RESPONSE": "",
  "CVV_RESPONSE": "",
  "CARD_BRAND_TRANSID": "",
  "REQUEST_API_VERSION": "4.14",
  "P3DS_VENDOR": "",
  "P3DS_RESPONSE": "",
  "PO_LI_ID_1": "1901896",
  "PO_LI_COUNT_1": 1,
  "PO_LI_AMOUNT_1": "0.01",
  "PO_LI_PROD_ID_1": "128917",
  "MBSHP_ID_1": "",
  "TRANS_NTOKEN_USED": 0
}
```


27.2 Reversals

27.2.1 Reversing a mandate (DBTREVERSE)

Merchants or customers can choose to cancel existing mandate whenever they want. To cancel an existing Mandate, Merchants should use DBTREVERSE Request_Action. This request_action also stops future rebills and cancels subscriptions.

27.2.2 DBTREVERSE transaction Request Example

```
https://api.inoviopay.com/payment/pmt_service.cfm
?request_action=DBTREVERSE&merch_acct_id=123456&site_id=456878&li_prod_id_1=14512&pmt_numb=
200125175423005031&li_value_1=0.01&request_currency=EUR&cust_fname=CustomerFN&cust_lname=Ma
sha&cust_email=msepauser%40gmail.com&bill_addr_country=AT&bill_addr_state=TW&bill_addr_city=Tri
erweiler&bill_addr=Unter%20den%20mystreetn%201&bill_addr_zip=54311&XTL_IP=11.11.11.11&request_api
_version=4.14&request_response_format=JSON&req_username=puser%40example.com&req_password=Te
stingpassword&argdebug=1&PROC_SUCCESS_URL=testingSuccess&PROC_ERROR_URL=testingError&re
quest_ref_po_id=18103694'
```

27.3 Refunds or Credits

27.3.1 Issuing refunds or credits (DBTCREDIT)

Merchants may request to credit a direct debit order by sending **DBTCREDIT** in the request_action parameter. Note that the request may not be processed successfully by the bank for certain transaction states (e.g. you cannot credit a payment that has not been collected by the bank).

To do a DBTCREDIT request, send the Order ID (PO_ID) of the original authorization in the **REQUEST_REF_PO_ID** parameter.

DBTCREDIT Required Fields
SITE_ID
REQ_USERNAME
REQ_PASSWORD
REQUEST_ACTION = DBTCREDIT
REQUEST_RESPONSE_FORMAT
REQUEST_REF_PO_ID
REQUEST_API_VERSION

27.4 Recurring transactions

EPS supports recurring payments through SEPA. Recurring payment Mandate Type authorizes subsequent rebills until canceled. Request Action **DBTDEBIT** is used for initial and subsequent rebills as authorized by Mandate. In summary these are the Recurring transactions scenarios.

- DBTDEBIT & request_rebill=2: start of rebill subscription
- DBTDEBIT & request_rebill=1 & cust_id & pmt_id: rebill payment #1
- DBTDEBIT & request_rebill=1 & cust_id & pmt_id: rebill payment #2
- DBTREVERSE: cancel mandate, end of subscription

27.4.1 Recurring transaction Request Example (DBTDEBIT & request_rebill=2)

```
https://api.inoviopay.com/payment/pmt_service.cfm
?request_action=DBTDEBIT&merch_acct_id=142376&site_id=116215&li_prod_id_1=128917&pmt_numb=30
01051720021110 31&li_value_1=0.01&request_currency=EUR&cust_fname=Masha&cust_lname=cust&cust_e
mail=customer.sepa%40example1.com&bill_addr_country=AT&bill_addr_state=TW&bill_addr_city=Trierw
eiler&bill_addr=Unter%20den%20mystreetn%201&bill_addr_zip=54311&XTL_IP=11.11.11.11&request_api_ver
sion=4.14&request_response_format=JSON&req_username=puser%40example.com&req_password=Oosii4
99&request_ref_po_id=18103687&request_rebill=2'
```

27.4.2 Recurring transaction Response Example (DBTDEBIT & request_rebill=2)

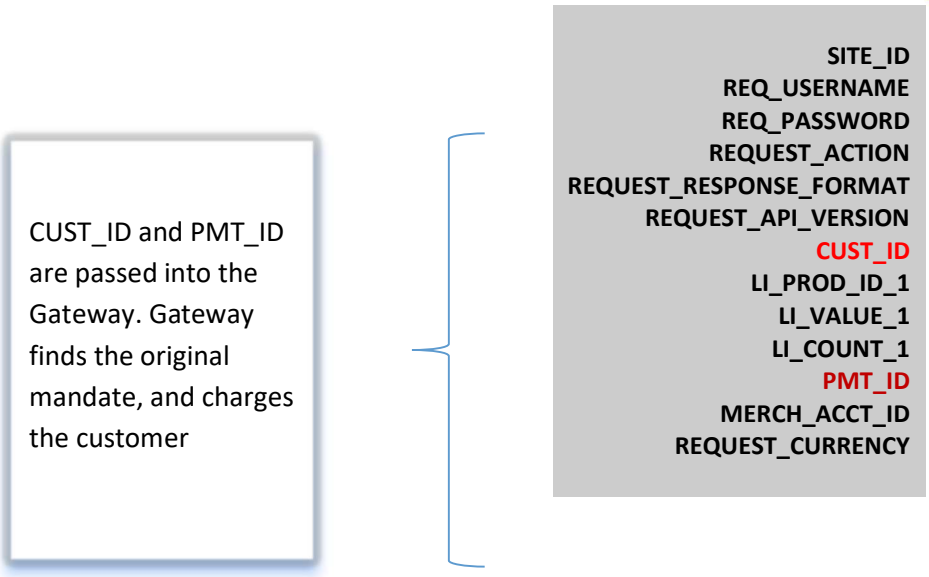

```
"REQUEST_ACTION": "DBTDEBIT",
"REQ_ID": "18785674",
"TRANS_STATUS_NAME": "PENDING",
"TRANS_VALUE": 0.01,
"CURR_CODE_ALPHA": "EUR",
"TRANS_VALUE_SETTLED": 0.01,
"CURR_CODE_ALPHA_SETTLED": "EUR",
"TRANS_EXCH_RATE": "",
```

```
  "TRANS_ID": 2001878902,
  "CUST_ID": 7534577,
  "XTL_CUST_ID": "",
  "PO_ID": 18103630,
  "XTL_ORDER_ID": "",
  "BATCH_ID": 140418,
  "PROC_NAME": "TrustPay",
  "MERCH_ACCT_ID": 142376,
  "DEBIT_TYPE ": "EPS",
  "CARD_TYPE": "",
  "CARD_PREPAID": "",
  "CARD_BANK": "",
  "CARD_DETAIL": "",
  "CARD_BALANCE": "",
  "PMT_L4": "5031",
  "PMT_ID": 4354240,
  "PMT_ID_XTL": "",
  "PMT_AAU_UPDATE_DT": "",
  "PMT_AAU_UPDATE_DESC": "",
  "PROC_UDF01": "",
  "PROC_UDF02": "",
  "PROC_AUTH_RESPONSE": "",
  "PROC_RETRIEVAL_NUM": "",
  "PROC_REFERENCE_NUM": "",
  "PROC_REDIRECT_URL": "https://aapi.trustpay.eu/SepaDirectDebitMandates/input/659b33f4-f5d8-4acb-
9dc7-6673e43e57ae",
  "AVS_RESPONSE": "",
  "CVV_RESPONSE": "",
  "CARD_BRAND_TRANSID": "",
  "REQUEST_API_VERSION": "4.14",
  "P3DS_VENDOR": "",
  "P3DS_RESPONSE": "",
  "PO_LI_ID_1": "1901856",
  "PO_LI_COUNT_1": 1,
  "PO_LI_AMOUNT_1": "0.01",
  "PO_LI_PROD_ID_1": "128917",
  "MBSHP_ID_1": "",
  "TRANS_NTOKEN_USED": 0
}
```


27.4.3 Recurring transaction Request Example (DBTDEBIT & request_rebill=1)


```
https://api.inoviopay.com/payment/pmt_service.cfm
?request_action=DBTDEBIT&merch_acct_id=142376&site_id=116215&li_prod_id_1=128917&pmt_numb=60
01000075433315031&li_value_1=0.01&request_currency=EUR&cust_fname=Mashada&cust_lname=Sam&cus
t_email=customer%40gmail.com&bill_addr_country=AT&bill_addr_state=TW&bill_addr_city=Trierweiler
&bill_addr=Unter%20den%20mystreet%201&bill_addr_zip=54311&request_api_version=4.14&request_resp
onse_format=JSON&req_username=apiuser%40example.com&req_password=Passwormed@98&request_r
ebill=1&CUST_ID=7534577&PMT_ID=4354240'
```


Recurring Transaction Fields



27.4.4 Recurring transaction Response Example (DBTDEBIT & request_rebill=1)


```
"REQUEST_ACTION": "DBTDEBIT",
"REQ_ID": "18787586",
"TRANS_STATUS_NAME": "APPROVED",
"TRANS_VALUE": 0.01,
"CURR_CODE_ALPHA": "EUR",
"TRANS_VALUE_SETTLED": 0.01,
"CURR_CODE_ALPHA_SETTLED": "EUR",
"TRANS_EXCH_RATE": "",
"TRANS_ID": 2001879023,
"CUST_ID": 7534577,
"XTL_CUST_ID": "",
"PO_ID": 18103688,
"XTL_ORDER_ID": "",
"BATCH_ID": 140418,
"PROC_NAME": "TrustPay",
"MERCH_ACCT_ID": 142376,
"DEBIT_TYPE": "EPS",
"CARD_TYPE": "",
"CARD_PREPAID": "",
"CARD_BANK": "",
"CARD_DETAIL": "",
"CARD_BALANCE": "",
"PMT_L4": "5031",
"PMT_ID": 4354240,
"PMT_ID_XTL": "",
"PMT_AAU_UPDATE_DT": "",
"PMT_AAU_UPDATE_DESC": "",
"PROC_UDF01": "",
"PROC_UDF02": "",
"PROC_AUTH_RESPONSE": "",
"PROC_RETRIEVAL_NUM": "5602944007",
"PROC_REFERENCE_NUM": "",
```

```
  "PROC_REDIRECT_URL": "",
  "AVS_RESPONSE": "",
  "CVV_RESPONSE": "",
  "CARD_BRAND_TRANSID": "",
  "REQUEST_API_VERSION": "4.14",
  "P3DS_VENDOR": "",
  "P3DS_RESPONSE": "",
  "PO_LI_ID_1": "1901910",
  "PO_LI_COUNT_1": 1,
  "PO_LI_AMOUNT_1": "0.01",
  "PO_LI_PROD_ID_1": "128917",
  "MBSHP_ID_1": "",
  "TRANS_NTOKEN_USED": 0
}
```


27.5 Additional Parameters

Field Name	Description
PROC_SUCCESS_URL	Used to indicate the URL to which the customer is redirected after successful payment (if PayNow was chosen)
PROC_ERROR_URL	Used to indicate the URL to which the customer is redirected after an error or failed payment (if PayNow was chosen)
PMT_NUMB	Used for IBAN Field (Required)
DEBIT_TYPE	Used to indicate the Debit type used for payment (SEPA, iDEAL, EPS) - Required

28 Testing

28.1 Using the test bank

Merchants may run test authorizations and other gateway actions using the test bank provided in the Payment Service Welcome Email.

If you did not receive your test bank information, contact your gateway support representative.

28.2 Basic Gateway Test

Use the table below as a reference for basic gateway testing scenarios and for testing your gateway response parser. The test bank should be used when running the tests.

NOTE: Transactions that have been blocked by the gateway due to invalid format or missing data will not be recorded and will not show up in merchant-facing reports on the Portal.

Case	Description	Expected Result
INVALID DATA	Send "1234" in PMT_EXPIRY	`API_ADVICE: Invalid Data`
REQUIRED FIELD	Send null in SITE_ID	`API_ADVICE: Required field` `REF_FIELD: SITE_ID`

INVALID MERCH_ACCT_ID	Send "9999" in MERCH_ACCT_FIELD	SERVICE_ADVICE: No merchant account configured
APPROVED AUTHORIZATION	Send "CCAUTHORIZE" in REQUEST_ACTION field along with valid data	TRANS_STATUS_NAME: APPROVED
APPROVED CAPTURE AUTHORIZATION	Send "CCCAPTURE" (follow this section for the required fields).	TRANS_STATUS_NAME: APPROVED
APPROVED SALE (AUTH + CAPTURE)	Send "CCAUTHCAP" in REQUEST_ACTION field along with valid data	TRANS_STATUS_NAME: APPROVED
EXPIRED CARD	Send "5.02" in LI_VALUE_1	SERVICE_ADVICE: Expired Card PROCESSOR_ADVICE: Expired Card
FAILED CVV	Send "5.03" in LI_VALUE_1	SERVICE_ADVICE: Failed CVV PROCESSOR_ADVICE: Failed CVV
FAILED AVS	Send "5.04" in LI_VALUE_1	SERVICE_ADVICE: Failed AVS PROCESSOR_ADVICE: Failed AVS
BANK DECLINED RESPONSE	Send "5.05" in LI_VALUE_1	SERVICE_ADVICE: Declined PROCESSOR_ADVICE: Declined
FRAUD	Send "5.06" in LI_VALUE_1	SERVICE_ADVICE: Fraud PROCESSOR_ADVICE: Fraud
OVER LIMIT	Send "5.07" in LI_VALUE_1	PROCESSOR_ADVICE: Overlimit
AVS NO MATCH	Send "5.08" in LI_VALUE_1	AVS_RESPONSE: N
MISSING REQUIRED FIELD	Send "5.18" in LI_VALUE_1	PROCESSOR_ADVICE: Missing Required Field
DOWNSTREAM PROCESSOR UNAVAILABLE	Send "5.30" in LI_VALUE_1	PROCESSOR_ADVICE: Downstream Processor Unavail
GENERAL DECLINE	Send "6.00" in LI_VALUE_1	PROCESSOR_ADVICE: General Decline
STOLEN CARD	Send "6.05" in LI_VALUE_1	PROCESSOR_ADVICE: Stolen Card
PICKUP CARD	Send "6.10" in LI_VALUE_1	PROCESSOR_ADVICE: Pickup Card
INVALID CVV	Send "6.20" in LI_VALUE_1	PROCESSOR_ADVICE: Invalid CVV
EXPIRED CARD	Send "6.24" in LI_VALUE_1	PROCESSOR_ADVICE: Expired Card
Timeout Void	Send "6.26" in LI_VALUE_1	SERVICE_ADVICE: Authorization has been voided in accordance to the merchant account Timeout Void settings. Please try transaction again

INSUFFICIENT FUNDS	Send “6.35” in LI_VALUE_1	PROCESSOR_ADVICE: Insufficient Funds
PARTIAL AUTH TEST	Send “PARTIAL_AUTH=1” “PARTIAL_AUTH_MIN=5.00” “LI_VALUE_1=”6.60”	TRANS_STATUS_NAME: APPROVED TRANS_VALUE:5.5
CONVERT CURRENCY	Send REQUEST_CURRENCY=EUR and LI_VALUE_1=7.00	TRANS_VALUE: 7 CURR_CODE_ALPHA: EUR TRANS_VALUE_SETTLED: 5.686415 CURR_CODE_ALPHA_SETTLED: USD TRANS_EXCH_RATE: 0.812345
CVV NO MATCH	Send “7.25” in LI_VALUE_1	CVV_RESPONSE:N
R1 REVOCATION OF ALL AUTHORIZATIONS	Send “8.81” in LI_VALUE_1	INDUSTRY_RESPONSE: R1 INDUSTRY_ADVICE: Revocation of All Authorizations
R3 REVOCATION OF ALL AUTHORIZATIONS	Send “8.83” in LI_VALUE_1	INDUSTRY_RESPONSE: R3 INDUSTRY_ADVICE: Revocation of All Authorizations
R1 REVOCATION OF ALL AUTHORIZATIONS	Send “9.91” in LI_VALUE_1	PROCESSOR_RESPONSE: R1 PROCESSOR_ADVICE: Revocation of All Authorizations1
R3 REVOCATION OF ALL AUTHORIZATIONS	Send “9.93” in LI_VALUE_1	PROCESSOR_RESPONSE: R3 PROCESSOR_ADVICE: Revocation of All Authorizations

28.3 Test Credit Cards

Use the credit cards below with any address, expiration date and CVV2 data.

Network	Credit Card Number
MASTERCARD	5105105105105100
MASTERCARD	5555555555554444
MASTERCARD	5546989999990033
VISA	4111111111111111
VISA	4907639999990022
AMERICAN EXPRESS	378282246310005
DINERS CLUB	38520000023237

DINERS CLUB	30569309025904
DISCOVER	6011111111111117
DISCOVER	6011000990139424
JCB	3530111333300000
JCB	3566002020360505
3D Secure	See 3D Secure

Appendix A: Service Request Types

REQUEST_ACTION	Description
ACHAUTHCAP	Used for authorization and capture requests.
ACHAUTHORIZE	Used for Authorizations without Capture
ACHREVERSE	Used for Authorization Capture Reversal
ACHCREDIT	Used for transaction credit requests.
APPLEPAYCONFIG	Instructs the endpoint to provide Apple Pay Configuration
CCAUTHORIZE	Used for sending transaction authorization-only requests.
CCCAPTURE	Used for sending transaction capture previous authorization requests.
CCAUTHCAP	Used for sending transaction “authorization and capture” requests.
CCREVERSE	Used for sending transaction reversal or void requests. Sending this will reverse the **original authorization**. For reversing CCCAPTURE transactions, merchants should use CCREVERSECAP as the request action.
CCREVERSECAP	Used for sending transaction reversal or void requests against a “CCCAPTURE” transaction.
CCCREDIT	Used for issuing transaction returns or credits.
CCRDR	Used for RDR Dispute Processing
CCRDRDELETE	Used for RDR Dispute Processing to show a case that has been removed by the customer/issuer
CCTC40	Used for TC40 Alerts Processing

CCSTATUS	Used for checking the status of a previous transaction or order.
CCTRANSUPDATE	Used to add receipts on transaction that was previously run (approved or declined)
DBTAUTHORIZE	Used for preparing Mandate without charging
DBTCAPTURE	Used to charge the Mandate for the submitted amount
DBTCREDIT	Used for SEPA Direct Debit Refund/Credit request
DBTDEBIT	Used for SEPA Direct Debit Pay Immediately 'Pay Now'
DBTREVERSE	Used for canceling existing mandate and end subscription
GOOGLEPAYCONFIG	Instructs the endpoint to provide Google Pay Configuration
TESTGW	Used for testing gateway availability.
TESTAUTH	Used for testing basic authentication.
SUB_CANCEL	Used for requesting cancelation of an active membership record.
SUB_UPDATE	Used for updating the Product ID of an existing membership record.
ACHAUTHCAP	Request a transaction for Electronic Funds Transfer
ACHAUTHORIZE	Authorize/Validate Check without funds transfer
ACHREVERSE	Reverse an ACHAUTHCAP
BOLETOAUTHCAP	Used for Brazilian Boleto Payment type
PIXSALE	Used for Brazilian Pix Payment type
PAGSALE	Used for Peru's PagoEfectivo Payment type

Appendix B: Transaction Status

TRANS_STATUS_NAME	Description
APPROVED	Transaction has been approved.

DECLINED	Transaction has been declined.
PENDING	Transaction is in pending status (expected on 3-D Secure, and preauthorization of online check transactions (i.e. Boleto, ACH, Pix etc.)).
RUNNING	Transaction processing is not completed or is waiting completion.
FAILED	Transaction did not finish payment completion (used in European Direct Debit transactions)

Appendix C: API Response Codes

API_RESPONSE	Description	Recommendation
100	Invalid login information (throttle)	Check your login credentials and try again. If you continue to receive this response, contact Client Support
101	Invalid login information	Check your login credentials and try again. If you continue to receive this response, contact Client Support
102	User not active	These credentials have been disabled. If you think this is an error, contact Client Support
103	Invalid site	The value of SITE_ID does not exist, or it does not match the authentication credentials provided.
104	Invalid service	Check the value of request_action to confirm it is correct.
105	Invalid service action	Check the value of request_action to confirm it is correct.
106	Invalid service object	Check the value of request_object to confirm it is correct.
110	Required field	A required key/value pair has not been included in the request. In the response, check the value of REF_FIELD to see what is missing
111	Invalid length	The length of a value is too short or long. Check the returned value of REF_FIELD to see which field may need editing
112	Not numeric	Numeric data is expected. Confirm the amount sent for LI_VALUE_x, which should only contain numerals and one decimal
113	Invalid Data	Something in the request was not expected. Check the values that were submitted for unusual characters, spaces, or null values where there perhaps should not be

115	Customer not found	If CUST_ID or CUST_ID_XTL was submitted, check these values and try again. If this response has come from a request without these parameters, contact Client Support
116	User MUST change password	User passwords expire every 90 days. This does not apply to API credentials.
118	New password must not match the previous 5 passwords	Try a different password.
119	request_ref_po_id and request_po_li_id mismatch	The order ID and the line item ID do not relate to one another. Check the order information.
120	System Error	Contact Client Support
125	Duplicate Login	This email address, a unique identifier, already exists.
130	Same Product ID found on different line items.	Check the values of LI_PROD_ID_x. Each one should have a unique ID. If the intent is to submit a purchase for multiples of the same product use LI_COUNT_x to indicate the quantity.
135	Duplicate Company Name	This company name is already in the system. If you are certain it doesn't already exist in the system, it could be a company with the same name, but doing business in a different region. Contact Client Support for assistance.
136	Duplicate Site Name	This site name already exists in our system.
150	Product Not Found	The product ID is not valid. It may not exist, or it might be associated with another site. Check the site ID and client ID
152	Product Type Not Found	The value for PROD_TYPE is not valid. Check the value and try again.
153	Duplicate XTL product id	This value is already in the system. To confirm and review, the ID can be searched for in our portal.
155	Selected currency not configured	Check the merchant account configuration in the portal. The MID's allowed currencies can be configured there.
160	Invalid product amount	Check the value of LI_VALUE_x to confirm it is the intended amount.
165	Currency not supported	Check the merchant account configuration in the portal. The MID's allowed currencies can be configured there. Additionally, check the value of PROCESSOR_RESPONSE in the

		response data. The MID made have a limitation.
170	Duplicate product amount and currency	A product with matching properties already exists within this Site
176	Duplicate product description and language	A product with matching properties already exists within this Site
180	Invalid transaction limit type	The limit type was not recognized. Try using the portal to adjust velocity settings.
181	Invalid limit type	The limit type was not recognized. Try using the portal to adjust velocity settings.
183	Payment Type is required	Confirm that PMT_TYPE has been submitted, and has not been included multiple times.
205	No Permissions on requested object	You may not be able to check and confirm your own user permissions, so it may be necessary for an administrator to check them for you. If you feel this is an error, contact your administrator or Client Support.
210	Merchant Account not found	Verify the value of MERCH_ACCT_ID
211	Currency not found	The expected format is three-character currency code.
215	Invalid Card Brand	Check the card brand submitted. If you are certain it's correct, contact Client Support
410	Field not supported with wallet payment	Check the value of REF_FIELD in the response to see what incompatible element was submitted in the request
411	REQUEST_CURRENCY mismatch with Cryptogram	The currency in the gateway request needs to match the currency that was packed into the ApplePay cryptogram

414	GooglePay token has expired

Appendix D: Service Response Codes

SERVICE_RESPONSE	Description
100	User Authorized
101	Service Available
102	Membership Updated
150	Product Not Found
152	Product Type Not Found
155	Selected currency not configured
157	MID has RDR Status OFF
190	Invalid Product Configuration
192	Product Not Active
200	CVV required by processor
201	Country required by processor
202	DOB required by processor
203	SSN required by processor
204	Address required by processor
205	City required by processor
206	State required by processor
207	Postal Code required by processor
208	Phone required by processor
209	IP required by processor
210	CPF required by processor

211	Email required by processor
212	FName required by processor
213	LName required by processor
215	Activity limit exceeded
216	Invalid amount
217	No such issuer
218	Wrong PIN entered
219	R0: Stop recurring payments
220	R1: Stop recurring payments
221	System malfunction
500	No merchant account configured
501	Customer not found
502	Transaction error
503	Service Unavailable
505	Order adjusted to zero
506	Capture amount exceeds order value
507	Order fully captured
510	Order already reversed
511	Order already charged back
512	Order not found
515	Order fully credited
516	Credit amount exceeds order value
518	Missing required field
520	Unsupported Currency

522	Unsupported card brand
525	Batch Closed: Please credit
526	ApplePay is not supported on this merch_acct_id
527	No ApplePay merch_acct_id configured
528	ApplePay MCC Restricted
530	Downstream Processor Unavailable
536	Order not settled: Please reverse
540	Maximum Auth Limit Exceeded
546	GooglePay MCC Restricted
547	No GooglePay merch_acct_id configured
548	GooglePay is not supported on this merch_acct_id
555	Call Center
560	Invalid Service Action
564	Invalid Terminal
565	Invalid Amount
570	Invalid Card Type
580	Unsupported Request
600	Declined
601	Scrub Decline
603	Fraud
605	Stolen Card
610	Pickup Card
615	Lost Card
620	Invalid CVV

621	Failed CVV
622	Invalid AVS
623	Failed AVS
624	Expired Card
625	Excessive Use
630	Invalid Card Number
635	Insufficient Funds
640	Retry
650	Do Not Honor
660	Partial Approval
670	Additional Authentication Required
675	Invalid Card Number, failed Mod 10 validation
680	Duplicate Transaction Detected
685	Duplicate Order Detected
690	Active Membership Exists
692	Invalid Rebill Product
695	Site Username Unavailable
697	Membership Not Active
698	Membership Not Found
699	Membership Not Set for Rebill
700	Scrub Decline
706	Failed Age Validation
707	Invalid CPF

Appendix E: Address Verification Service Codes

Code	Description	Credit Card Network
A	Street address matches, but 5-digit and 9-digit postal code do not match.	Standard domestic (US)
B	Street address matches, but postal code not verified.	Standard international
C	Street address and postal code do not match.	Standard international
D	Street address and postal code match. Code "M" is equivalent.	Standard international
E	AVS data is invalid or AVS is not allowed for this card type.	Standard domestic (US)
F	Card member's name does not match, but billing postal code matches.	American Express only
G	Non-U.S. issuing bank does not support AVS.	Standard international
H	Card member's name does not match. Street address and postal code match.	American Express only
I	Address not verified.	Standard international
J	Card member's name, billing address, and postal code match.	American Express only
K	Card member's name matches but billing address and billing postal code do not match.	American Express only
L	Card member's name and billing postal code match, but billing address does not match.	American Express only
M	Street address and postal code match. Code "D" is equivalent.	Standard international
N	Street address and postal code do not match.	Standard domestic (US)

O	Card member's name and billing address match, but billing postal code does not match.	American Express only
P	Postal code matches, but street address not verified.	Standard international
Q	Card member's name, billing address, and postal code match.	American Express only
R	System unavailable.	Standard domestic (US)
S	Bank does not support AVS.	Standard domestic (US)
T	Card member's name does not match, but street address matches.	American Express only
U	Address information unavailable. Returned if the U.S. bank does not support non-U.S. AVS or if the AVS in a U.S. bank is not functioning properly.	Standard domestic (US)
V	Card member's name, billing address, and billing postal code match.	American Express only
W	Street address does not match, but 9-digit postal code matches.	Standard domestic (US)
X	Street address and 9-digit postal code match.	Standard domestic (US)
Y	Street address and 5-digit postal code match.	Standard domestic (US)
Z	Street address does not match, but 5-digit postal code matches.	Standard domestic (US)

Appendix F: CVV Response Codes

Code	Description
M	Match
N	No Match
P	Not Processed

S	Not Supported
U	Service Not Available
X	No CVC/CVV/CVV2/CID Response Data Available
(empty)	No CVC/CVV/CVV2/CID Response Data Available

Appendix G: Negative Option Billing- MCC 5968

Effective April 12, 2019, all MasterCard acquirers and processors are required to ensure their merchants which use Negative Option Trial Continuity billing models (MCC 5968) providing physical products are compliant with revised standards published in AN 2202.

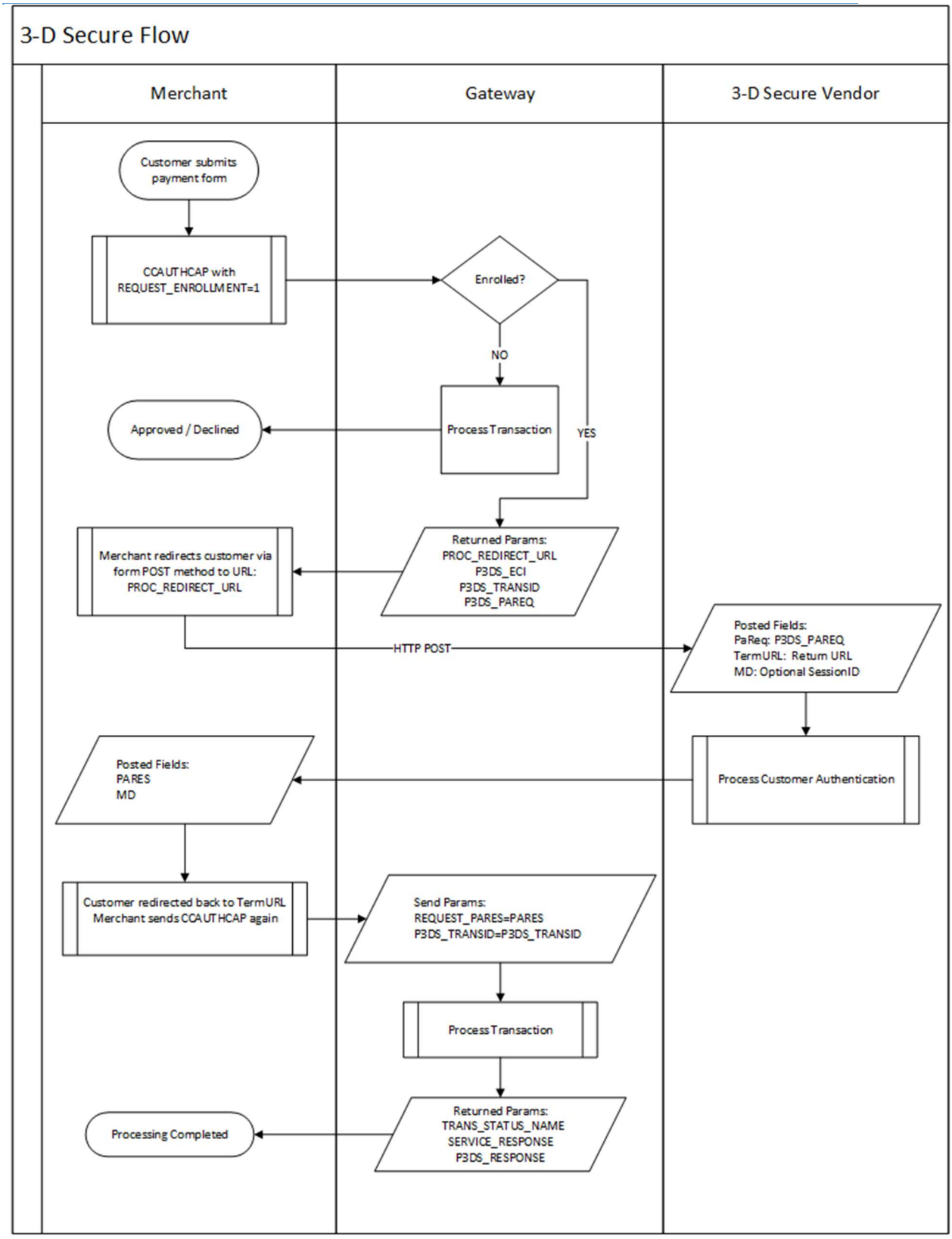
The gateway provides functionality and accepts data in support of these standards, and this appendix provides guidance on how the gateway API should be used for this billing model and which parameters are relevant.

Please note that this functionality may only be in place for some ISOs. Merchants should contact their gateway support representative for additional information on whether this section is relevant to them.

Field Name	Description
REQUEST_ACTION	Use the "CCTRANSUPDATE" value to add a receipt to a transaction that was previously run. Because MasterCard's guidelines state that a customer receipt must be sent in both the case of an authorized rebill OR a declined rebill, it is not possible to know the exact content of the receipt at the time of the authorization request (since a receipt for an authorization will be much different from a receipt for a decline). Therefore, this action is necessary to update the order with the appropriate customer receipt AFTER the authorization request has been completed.
MBSHP_ID_XTL	Customer Membership ID. Use this parameter to identify a specific subscription to a customer.
TRANS_REBILL_TYPE	Use this parameter with CCAUTHCAP and CCAUTHORIZE to indicate one of the following conditions: "NONE – send this value if the authorization is not related to a subscription. Valid CVV will be required. (note that a "Card on File" transaction without CVV is not supported for MC 5968 Trial accounts and will be declined)

	"TRIAL" – send this value if the authorization is for the start of a trial-based subscription. A new, unique MBSHP_ID_XTL value and a valid CVV will be required. "INITIAL" – Send this value if the authorization is for the start of a NON-trial-based subscription. A new, unique MBSHP_ID_XTL value and a valid CVV will be required. "REBILL" – send this value if the authorization is for the continuation of an existing subscription. An existing MBSHP_ID_XTL is required. If the rebill follows a trial, then TRANS_TRIAL_REBILL_CUSTOMER_CONSENT is required. It is required that any previous rebills MUST have been updated with a TRANS_CUSTOMER_RECEIPT. First transaction after trial with subsequent transactions possible as rebill/membership service , REBILL-Rebill transaction with prior transaction of Initial type previously captured)
TRANS_CUSTOMER_RECEIPT	The entire URL encoded contents of the receipt which merchant transmitted to the customer. Use REQUEST_ACTION=CCTRANSUPDATE to record this parameter after the authorization attempt is completed. This will be used to store the receipt for a successful authorization OR the receipt for an unsuccessful authorization. This is required prior to the next subsequent rebill or else the attempt for the next rebill will be rejected.
TRANS_TRIAL_REBILL_CUSTOMER_CONSENT	This is to store Customer Consent for subsequent rebill after a trial. This should be sent with the authorization request for that rebill and is required.
TPPE_ID	ID of Third-Party Processing Entity (e.g. CRM, eCommerce Platform, etc.) which is transmitting the request to the gateway on the merchant's behalf. Required when merchant is not transmitting cardholder data directly from their own systems.

Appendix H: 3-D Secure Flow



Appendix I: Automatic Account Updater (AAU)

In the event there is a change on Credit Card on record that might impact future transactions, AAU-enrolled Merchants will not experience approval changes caused by Credit card number changes. Changes like a new expiration date, account status change or a new account number will be automatically updated before there's any effect on future transactions.

When a Card is updated with a new expiration date, the new expiration date, Update date and description will be reported. Occasionally, since Credit Card numbers are encrypted/tokenized, a new Credit Card will only be reported by an Update Date and Description. Where available, the Last 4 digits will be updated but the PMT_ID remains the same. The table below outlines different scenarios and their respective examples. For more information regarding this service, contact our Client Services.

Update Response Action	Card Type	Description/ Element Name
Closed Account Advise	VISA	(**PMT_AAU_UPDATE_DESC: "**Closed Account Advise**", PMT_AAU_UPDATE_DT: "**2019-07-24**"**)
Expiration Date Change Message	VISA	**(PMT_AAU_UPDATE_DESC:** Expiration Date Change Message**", PMT_EXPIRY -** "012022", **PMT_AAU_UPDATE_DT: "**2019-07-24**"**)
Account Number Change Message	VISA.	(**PMT_AAU_UPDATE_DESC**: "Account Number Change Message"**, PMT_EXPIRY:** "012022", **PMT_L4:**"6183", **PMT_AAU_UPDATE_DT: "**2019-07-24**"**)
Match Made, Account Closed	MasterCard	(**PMT_AAU_UPDATE_DESC:** "Match Made, Account Closed", **PMT_AAU_UPDATE_DT: "**2019-07-24")
Match Made, Expiration Date Changed	MasterCard	(**PMT_AAU_UPDATE_DESC: "** Match Made, Expiration Date Changed", **PMT_EXPIRY:** "012022"**, PMT_L4**:"1860", **PMT_AAU_UPDATE_DT: "**2019-07-24"**)**
Match Made, Update Data Provided	MasterCard	(**PMT_AAU_UPDATE_DESC:"** Match Made, Update Data Provided"**, PMT_EXPIRY:"** 012022"**, PMT_L4**:"1920", **PMT_AAU_UPDATE_DT:** 2019-07-24"**)**

Appendix J: Visa Trial Processing

Effective April 18, 2020, Visa has updated rules for merchants that offer free trials or introductory offers as part of an ongoing subscription service.

The gateway provides functionality and accepts data in support of these rules, and this appendix provides guidance on how the gateway API should be used for this billing model and which parameters are relevant.

Please note that this functionality may not function properly for all processors (e.g. dynamic descriptor support may not allow trial purchases to be identified). Merchants should contact their gateway support representative with any questions.

Field Name	Requirement	Description
MBSHP_ID_XTL	OPTIONAL	Customer Membership ID. Use this parameter to identify a specific subscription to a customer.
TRANS_REBILL_TYPE	OPTIONAL (Unless Transaction Type is "TRIAL")	Use this parameter with CCAUTHCAP and CCAUTHORIZE to indicate one of the following conditions: "NONE"– send this value if the authorization is not related to a subscription. Valid CVV will be required "TRIAL" – send this value if the authorization is for the start of a trial-based subscription. A new, unique MBSHP_ID_XTL value and a valid CVV will be required. "INITIAL" – Send this value if the authorization is for the start of a NON-trial-based subscription. A new, unique MBSHP_ID_XTL value and a valid CVV will be required. "REBILL" – send this value if the authorization is for the continuation of an existing subscription. An existing MBSHP_ID_XTL is required.
TRANS_CUSTOMER_RECEIPT	OPTIONAL	The entire URL encoded contents of the receipt which merchant transmitted to the customer.

Appendix K: Card on File Matrix

A Card on File (*Credential on File, COF or Stored Credential*) is payment information like an account number or payment token that is stored by a merchant or its agent to process future purchases for a cardholder. Future purchases may include Installments, subscription rebills and occasional one time purchases initiated by the Merchant (MIT) or the Client (CIT).

The table below outlines the new parameters mandated and added to our gateway for Card on File transactions.

New Parameter	Values	Description
REQUEST_INITATOR	CIT (C) and MIT (M)	Used to identify transaction origination

REQUEST_INSTALLMENT	O (not used), 1 (used)	Used to denote if the rebill is an installment with start and end date.
PMT_NUMB_COF	O (not used), 1 (used)	Used to denote use of payment number stored by Merchant
REQUEST_REBILL	1 (yes for rebill), 2(Start Subscription), not used	Used to denote if the transaction is subscription card storage, a rebill or not used at all