- Version
  - History
- Introduction
- Audience
- Authentication
- URLs
  - Server API
  - WPF
  - Consumer API
  - Tokenization API
  - Genesis KYC Services JSON API
  - Transaction API
- Transactions
  - Invoking a Transaction
  - Card
    - Recurring Transactions
      - Init Recurring Sale
      - Recurring Sale
    - Authorize
    - Capture
    - Sale
    - Argencard
    - Aura
    - Bancontact
    - Cabal
    - Cencosud
    - Credit (CFT)
    - Elo
    - Naranja
    - Nativa
    - Tarjeta Shopping
  - Non-financial Transactions
    - Account Verification
  - 3DS Card
    - Authorize 3D
    - Sale 3D
    - Init Recurring Sale 3D
    - Notification for Asynchronous Payments
  - Wallets
    - Neteller
    - PayPal Express Checkout
    - QIWI
    - WebMoney
    - Wechat
    - Alipay
    - eZeeWallet
  - Vouchers
    - CashU
    - Neosurf
    - Paysafecard
  - Online Banking ePayments (oBeP)
    - Banco do Brasil
    - Bancomer
    - Bradesco
    - EPS
    - GiroPay
    - iDebit
      - Payin
      - Payout
    - iDeal
    - InstaDebit
      - Payin
      - Payout
    - InstantTransfer
    - Itau
    - Multibanco
    - MyBank
    - Online Banking
      - Payment Types
      - Bank Codes
    - P24
    - PayU
    - POLi
    - PSE (Pagos Seguros en Linea)
    - RapiPago
    - SafetyPay

    - Santander
    - SDD Init Recurring Sale
    - SDD Recurring Sale
    - SDD Refund
    - SDD Sale
    - Sofort
    - Trustly Sale
    - TrustPay
    - UPI
    - Webpay
    - Davivienda
- Cash Payments
    - Baloto
    - Banco de Occidente
    - Boleto
    - Efecty
    - OXXO
    - Pago Facil
    - Redpagos
- Gift Cards
    - Intersolve
    - Fashioncheque
    - TCS
    - Split Payments
- Invoice Payment Methods
    - Klarna Authorize
    - Klarna Capture
- Crypto
    - BitPay Sale
- Payouts
    - Bank Pay-out
        - Bank Names
    - BitPay Payout
    - eZeeCard Payout
    - Payout
    - Money transfer Payout
    - Non-money transfer Payout
    - SCT Payout
    - Trustly Withdrawal
    - African Mobile Payout
    - TransferTo Payout
- Mobile Payments
    - Apple Pay
    - African Mobile Sale
- Reversals
    - Refund
    - Async Refund
    - Void
    - Klarna Refund
    - BitPay Refund
    - Partial Reversal
- PayByLink
    - Workflow
    - Combination with Pay Later functionality
    - Reminders configuration
- Alternative Payment Method External Events
    - Introduction
    - List of external events per alternative payment method
    - Email Notification
    - API Notification
- Advanced risk management with RiskParams
- Credential On File (COF)
- Currency and Amount Handling
- Dynamic Descriptor
- Electronic Commerce Indicator
- Issuer Response Codes
- Manually Reviewed Transactions
- Partial Approvals
- Preauthorizations
    - Preauthorization
        - Visa
        - Mastercard
        - Maestro
    - Incremental Authorize
    - Capture
    - Full Reversal
    - Partial Reversal
- Required vs Optional API params
- Transaction States
- Supported Card Brands
- Document ID Parameter
- Business Attributes

- Reconcile
  - Single Transaction
    - Preauthorization
  - By date range
- Payment Authorizations
  - Payment Authorizations API
  - By date range
- Processed Transactions
  - Processed Transaction API
  - Single Processed Transaction
  - By date or post date range
- Processed Batches
  - Processed Batches API
  - By date or post date range
- Chargebacks
  - Chargeback reversals
  - Chargeback notifications
  - Chargeback API
  - Single Chargeback
  - List of Chargebacks
  - By date range
  - Chargeback types
- Retrieval Requests
  - Retrieval Request notifications
  - Retrieval Request API
  - Single Retrieval Request
  - List of Retrieval Request
  - By date range
- Funding Accounts
  - Funding Accounts API
    - By Post Date
- Fraud reports
  - SAFE/TC40 API
  - Single SAFE/TC40 report
  - List of SAFE/TC40 report
  - By date range
- Blacklists
  - Invoking a Request
  - Response
- Asynchronous Transactions and Notifications
  - Asynchronous Transactions
  - Notifications
- WPF
  - Workflow
  - WPF API
    - URLs
    - Create
    - Notification
    - Reconcile
    - Reconcile by Date range
    - Errors
  - WPF transaction types
  - WPF Internationalization (i18n)
- 3DSecure
  - V1
    - Introduction
    - Authentication Flow
    - Testing
  - V2
    - Introduction
    - Request Params
      - Control Params
      - Recurring Params
    - Authentication Flows

        - Frictionless
        - Frictionless with 3DSecure Method
        - Challenge
        - Challenge with 3DSecure Method
        - Fallback
        - Fallback with 3DSecure Method
    - Testing

- Authentication Services
  - Introduction
    - SCA with 3DSecure
    - Alternative services
- Genesis KYC Services
  - General Info
  - Create Consumer Registration
  - Update Consumer Registration
  - Create Transaction
  - Update Transaction
  - Identity Document Upload
  - Identity Document Download
  - Make call
  - Update call
  - Error Response
  - Kyc Service Notification
- Genesis Fx Services
  - General Info
  - Get Tiers
  - Get Tier
  - Get Rates
  - Get Rate
  - Search Rate
- Consumers
  - Introduction
  - Consumer API
    - Create consumer
    - Retrieve consumer
    - Update consumer
    - Disable consumer
    - Enable consumer
    - Get consumer cards
- Tokenization
  - Introduction
  - Tokenization API
    - Accepted cardholder parameters
    - Consumer required
    - Tokenize
    - Detokenize
    - Update token
    - Validate Token
    - Delete Token
    - Get card
  - How to tokenize cardholder data in Processing API
    - Create a Consumer
    - Use existing consumer
  - How to use tokens in Processing API
    - Requests
  - How to tokenize cardholder data in WPF API
    - Create a Consumer
    - Use existing consumer
  - How to use tokens in WPF API
  - Supported transaction types
  - Importation of external tokens and card details
    - CSV file format
    - Encryption of the CSV file
    - Uploading the encrypted CSV file to remote SFTP server
    - Downloading a response CSV file from the remote SFTP server
- Transaction API
  - Transaction Card Expiry Date Update API
- APM Services
  - Alipay Register Merchant
  - Introduction
    - Alipay Register Merchant Site info parameters

- Site info
- Klarna
- Introduction
  - Release Remaining Authorization API
  - Resend Invoice API
  - Update Order Items API
  - Update Order Address API
- Trustly Select Account
- Introduction
- Trustly Register Account
- Introduction
- TransferTo
- Introduction
  - Retrieve Payers API

- Errors
  - Error groups table
  - Error codes tables

- Client Integrations
- Client-side encryption
  - Client side
    - Prevent form submit action
    - JavaScript only
  - Server side

- Shopping Carts
- Testing
  - 3DS v1
  - 3DS v2

- Status Page
- Infrastructure and Uptime
- Penetration Testing Warning
- AVS Status Codes
- Level 3 Travel Data
  - Travel Types
    - Airline Itinerary Data (AID)
    - Car Rental
    - Hotel Rental
    - Ancillary Charges
    - Miscellaneous Charges
  - Special Cases
    - Travel Authorize (3D) and Capture
    - Visa Refund
  - Allowed Values
    - Car Rental Classes
    - Charge Types
    - Car Rental Extra Charges
    - Hotel Rental Extra Charges
    - Ticket Change Indicators
    - Credit Reason Indicators

- Genesis SCA Services
  - General Info
  - SCA Checker


# Version

**API version 22.8**

**Documentation version 22.8**

## History

| Version | Date | Name | Description |
|---|---|---|---|
| 1.0 | 2011/06/10 | Emil Petkov | Initial version |
| 1.1 | 2011/11/11 | Emil Petkov | Added optional shipping params section for the shipping address |
| 1.2 | 2012/04/18 | Emil Petkov | Added dynamic descriptor functionality |

| Version | Date | Name | Description |
|---|---|---|---|
| 1.3 | 2012/08/03 | Emil Petkov | Added new transaction type AVS and Account Verification |
| 2.0 | 2013/03/29 | Emil Petkov | Added gaming and MOTO flags and support |
| 2.1 | 2013/04/11 | Emil Petkov | Split Credits with an initial ref transaction and Payouts without a reference transaction |
| 2.2 | 2013/05/26 | Emil Petkov | Added new transaction type InitRecurringSale3D - recurring payments with initial 3D |
| 2.3 | 2014/01/15 | Emil Petkov | Removed the transaction types DebitSale and IdealSale |
| 2.4 | 2014/01/22 | Emil Petkov | Added authorization code and issuer response code to API, section for issuer response codes |
| 2.5 | 2014/01/29 | Emil Petkov | Moto and gaming flags are now returned in the transaction response if present/marked in the request |
| 2.6 | 2014/02/05 | Emil Petkov | Dynamic descriptor params are now returned in the transaction response if present in the request |
| 2.7 | 2014/02/15 | Emil Petkov | Recurring advices are now returned in the transaction response if received from the issuer |
| 2.8 | 2014/05/01 | Emil Petkov | Added required vs. optional API params |
| 2.9 | 2014/06/16 | Emil Petkov | Currency handling rework and API description (different currency exponents, etc) |
| 3.0 | 2014/08/28 | Emil Petkov | Added risk related APIs - chargebacks, retrieval requests, blacklists |
| 3.1 | 2014/11/25 | Emil Petkov | Added API support for partial approvals |
| 3.2 | 2014/12/03 | Emil Petkov | Added WPF I18N specifics |
| 3.3 | 2014/12/08 | Emil Petkov | Added shopping carts and client integrations list |
| 3.4 | 2014/12/16 | Hristo Tanchev | Now the remote ip can be either a required or optional API param |
| 3.5 | 2015/02/22 | Dimitar Kostov | Added API for eZeeWallet and PayByVoucher via Yeepay |
| 3.6 | 2015/02/23 | Hristo Tanchev | Added API for CashU and Paysafecard |
| 3.7 | 2015/03/13 | Dimitar Kostov | Added API for Sofort |
| 3.8 | 2015/03/18 | Hristo Tanchev | Added API for PPRO |
| 3.9 | 2015/04/15 | Emil Petkov | New WPF API and WPF payment methods |
| 4.0 | 2015/05/14 | Hristo Tanchev | Added API for Neteller and ABN iDEAL |
| 4.1 | 2015/06/15 | Dimitar Kostov | Added WPF custom attributes - bin, tail |
| 4.2 | 2015/08/10 | Tsvetelina Borisova | Added WPF custom attributes - default, expiration date |
| 4.3 | 2015/08/12 | Vladimir Korichkov | Added API for WebMoney and POLi |
| 4.4 | 2015/08/19 | Tsvetelina Borisova | Added fraud related API for TC40/SAFE (fraud reports) |
| 4.5 | 2015/08/27 | Emil Petkov | 3D attributes xid and cavv are now not re- quired in the MPI sync attempted only workflow, only eci is |
| 4.6 | 2015/09/02 | Emil Petkov | New transaction type PayByVoucher Sale for purchasing vouchers via credit cards. Reworked the PayByVouchers section |
| 4.7 | 2015/09/04 | Emil Petkov | Added penetration testing warning for merchants |
| 4.8 | 2015/09/06 | Emil Petkov | Reconcile API now works with ARN and transaction ID in addition to unique ID |
| 4.9 | 2015/09/11 | Hristo Tanchev | Added Electronic Commerce Indicator to notifications |
| 5.0 | 2015/09/30 | Vladimir Korichkov | Added API for INPay |
| 5.1 | 2015/12/09 | Pepa Simeonova | Added event parameter to notifications for fraud transactions. |
| 5.2 | 2016/02/02 | Hristo Tanchev | Added API for P24 |
| 5.3 | 2016/02/16 | Tsvetelina Borisova | Added ARN in reconcile response if available. |
| 5.4 | 2016/03/18 | Tsvetelina Borisova | Added page for Manually Reviewed Transactions |
| 5.5 | 2016/03/17 | Tsvetelina Borisova | Added API for SDD. |
| 5.6 | 2016/03/28 | Emil Petkov | Extended PayByVouchers processing and WPF APIs with redeem type and card type |
| 5.7 | 2016/04/08 | Emil Petkov | Added info for statuspage.io, uptime and infrastructure, and new shopping carts |
| 5.8 | 2016/05/11 | Tsvetelina Borisova | Change the example for chargeback API - amount is returned in minor currency unit |

| Version | Date | Name | Description |
|---|---|---|---|
| 5.9 | 2016/05/22 | Emil Petkov | Now birth date API param is required only for Visa initial transactions with Financial Service MCCs (e.g. 6012) |
| 6.0 | 2016/06/30 | Pepa Simeonova | Added dynamic descriptor params to WPF payments. |
| 6.1 | 2016/07/20 | Stefan Slaveykov | Now usage can be either a required or optional API param |
| 6.2 | 2016/07/21 | Tsvetelina Borisova | Update documentation for chargebacks API |
| 6.3 | 2016/12/08 | George Naydenov | Added API for Trustly |
| 6.4 | 2017/01/20 | Plamen Terziev | Added AVS Codes |
| 6.5 | 2017/02/16 | Nikolay Petrov | Added API for PayPal Express Checkout |
| 6.6 | 2017/03/20 | Iskar Enev | Added API for Citadel |
| 6.7 | 2017/03/30 | Stanislav Mihailov | Added API for Instadebit/iDebit |
| 6.8 | 2017/03/31 | Tsvetelina Borisova | Added API for SCT Payout |
| 6.9 | 2017/05/12 | Nikolay Petrov | Added reference transaction unique id to the Reconcile API response |
| 7.0 | 2017/05/13 | Nikolay Petrov | Added API for Earthport |
| 7.1 | 2017/05/15 | Stanislav Mihailov | Added API for Wechat |
| 7.2 | 2017/05/30 | George Naydenov | Added API for Alipay |
| 7.3 | 2017/06/05 | Hristo Tanchev | Added API for PaySec |
| 7.4 | 2017/06/06 | Hristo Tenev | Added card brand and card number tags in reconcile response. |
| 7.5 | 2017/06/27 | Lukasz Wojcik | Added lifetime for WPF Payments. |
| 7.6 | 2017/07/06 | Nikolay Petrov | Removed AVS transaction type. AVS response code and text are returned in the transactions responses when present and supported or in the notifications for payments with asynchronous workflow. |
| 7.7 | 2017/07/06 | Iskar Enev | Added APM External Events. |
| 7.8 | 2017/08/11 | Samuil Goranov | Added support for optional shipping address params in Processing and WPF APIs. |
| 7.9 | 2017/08/17 | Lukasz Wojcik | Remove remote ip parameter for transactions with reference. It will be copied from reference transaction |
| 8.0 | 2017/08/18 | Lukasz Wojcik | Added invalid transaction types for amount parameter to WPF response. |
| 8.1 | 2017/08/18 | Emil Kirilov | Added search option by import date in Chargeback, Retrieval request and Fraud APIs. |
| 8.2 | 2017/09/05 | George Naydenov | Added API for RPN Payment |
| 8.3 | 2017/10/13 | Ventsislav Dimitrov | Added API for FashionCheque gift card and split payments. |
| 8.4 | 2017/10/13 | Pepa Simeonova | Added API for Intersolve gift card and split payments. |
| 8.5 | 2017/10/13 | Stanislav Mihailov | Added API for TCS gift card and split payments. |
| 8.6 | 2017/10/23 | Iskar Enev | Added Authentication Services and iSignThis. |
| 8.7 | 2017/11/02 | George Naydenov | Added API for RPN Payout |
| 8.8 | 2017/11/24 | Hristo Tenev | Added API for Paycommerce |
| 8.9 | 2017/11/25 | George Naydenov | API update with new sections Card, 3DS Card, Common. Removing older payment methods. |
| 9.0 | 2017/12/18 | George Naydenov | Added API for Neosurf. |
| 9.1 | 2018/01/02 | George Naydenov | Removed API for Inpay, ABN iDeal, Teleingreso and PayByVoucher. |
| 9.2 | 2018/01/03 | Iskar Enev & Nikolay Petrov | Added Tokenization service. |
| 9.3 | 2018/01/12 | Hristo Tanchev | Added API for Klarna. |
| 9.4 | 2018/01/18 | Stefan Petrov | Added ARN in WPF reconcile response if available. |
| 9.5 | 2018/01/26 | George Naydenov | Added API for Astropay Direct, Pago Facil, Link, Carulla, Davivienda. |
| 9.6 | 2018/02/22 | George Kostov & Hristo Tenev | Added API for Genesis KYC Services. |
| 9.7 | 2018/02/22 | Emil Kirilov | Added API for PSE. |
| 9.8 | 2018/02/26 | Stefan Petrov | Added API for RapiPago, Webpay, Banco de Chile. |

| Version | Date | Name | Description |
|---|---|---|---|
| 9.9 | 2018/03/01 | Stefan Petrov | Added API for Surtimax, Efecty, Cabal, Cencosud, Hipercard, Elo, Aura, Itau, Bradesco, Tarjeta Shopping, BBVA Bancomer, Boleto, Redpagos, Emprese De Energia, GiroPay, InstantTransfer, Multibanco. |
| 10.0 | 2018/03/02 | George Naydenov | Added API for OXXO, Argencard, Naranja, Nativa, Cartao Mercado Livre, Astropay Card, Banamex, Santander, Santander Cash, Zimpler PayU. |
| 10.1 | 2018/03/02 | Emil Kirilov | Added API for Baloto, Banco de Occidente, Banco do Brasil. |
| 10.2 | 2018/04/10 | Stefan Petrov | Added API for Entercash. |
| 10.3 | 2018/05/08 | Hristo Tenev | Added API for eZeeWallet Payout. |
| 10.4 | 2018/05/31 | Maya Nedyalkova | Added 3D MasterCard test cards. |
| 10.5 | 2018/05/31 | George Naydenov | Added API for QQPay. |
| 10.6 | 2018/06/22 | Hristo Tanchev | Added API for Credential on File (COF). |
| 10.7 | 2018/07/05 | George Naydenov | Removed Paysec and added Online Banking OBeP. |
| 10.8 | 2018/07/18 | Stanislav Mihailov | Added the API param issuer_oct_enabled to Visa-based Account Verification transactions, to allow merchant to verify if issuer supports OCTs for the given PAN |
| 10.9 | 2018/07/27 | Yordan Pulov | Added Money Transfer support to Payouts. |
| 11.0 | 2018/08/01 | George Naydenov | Removed Paysec Payout and added Bank Payout. |
| 11.1 | 2018/08/02 | Stanislav Mihailov | Remove API for Citadel |
| 11.2 | 2018/08/31 | Hristo Tenev | Added API for BitPay. |
| 11.3 | 2018/08/31 | Maya Nedyalkova | Add rc_code and rc_description in notification parameters. |
| 11.4 | 2018/10/10 | Ralitsa Borisova | Remove Paycommerce, RPN Payment, Link, Davivienda, Banco de Chile, Cartao Mercado Livre documentation. |
| 11.5 | 2018/10/11 | Emil Petkov | Introduced debt repayments - now birth date API param is required also for Mastercard/- Maestro initial transactions with Financial Service MCCs (e.g. 6012), where merchant is UK-based, transaction is domestic (with UK-based bin), and card type is DEBIT. |
| 11.6 | 2018/11/02 | Yordan Pulov and Stanislav Mihailov | Introduced Travel layer and Added Level 3 Travel API. |
| 11.7 | 2018/11/21 | Hristo Tenev | Added Payout support for BitPay |
| 11.8 | 2018/12/05 | Stefan Petrov | Added API for eZeeCard Payout |
| 11.9 | 2018/12/05 | Nikolay Petrov | Added Pay Later support to the WPF. Introduced Reminders module. |
| 12.0 | 2019/01/07 | Yordan Pulov | Changed Contract Merchant Category Codes for Level 3 Visa Car and Hotel Rental Transactions. |
| 12.1 | 2019/01/31 | Aleksandar Krastev | Introduced Consumer API. Extended Tokenization API to require a consumer. Extended WPF and Processing APIs to create consumers and tokenize card details in one step. WPF API can use saved cards to make payments. |
| 12.2 | 2019/02/18 | Yordan Pulov | Extend the Online Banking bank codes and add payment type. |
| 12.3 | 2019/02/26 | Stefan Petrov | Added API for Tola payments. |
| 12.4 | 2019/03/13 | Stanislav Mihailov | Added support for new non-money transfer payout types. |
| 12.5 | 2019/03/19 | Nikolay Valchanov | Added cardholder and expiration dates params to reconcile APIs. |
| 12.6 | 2019/03/19 | Vladislav Yakimov | Added retrieve endpoint for the Consumer API. |
| 12.7 | 2019/03/22 | Ventsislav Dimitrov | Added support for Preauthorizations. |
| 12.8 | 2019/03/28 | Nikolay Petrov | Added support for importation of external tokens and card details. |
| 12.9 | 2019/04/01 | Nikolay Petrov & Yasen Angelov | Added support for Pay by Link functionality. |
| 13.0 | 2019/04/09 | Hristo Tenev | Added API for transaction card expiry date update. |
| 13.1 | 2019/04/11 | Stanislav Mihailov | Added API support for purchase of cryptocurrency transactions. |
| 13.2 | 2019/04/12 | George Naydenov | Removed QQPay transaction type. |
| 13.3 | 2019/04/15 | Nikolay Valchanov | Added usage and description params to WPF reconcile API. |
| 13.4 | 2019/04/22 | Nikolay Valchanov | Added optional bic param to iDeal transactions. |
| 13.5 | 2019/05/31 | George Naydenov | Added Genesis Fx Services. |
| 13.6 | 2019/06/07 | Rumen Milushev | Added WPF API Reconcile by_date. |
| 13.7 | 2019/06/14 | Yordan Pulov | Extended the Online Banking bank codes and Bank Payout banks. |

| Version | Date | Name | Description |
|---|---|---|---|
| 13.8 | 2019/07/02 | Aleksandar Krastev | Added Tokenization API get masked card details for token. |
| 13.9 | 2019/07/04 | Vladislav Yakimov | Added API for TransferTo Payout and TransferTo Payers retrieve. |
| 14.0 | 2019/07/09 | Stefan Petrov | Extend the Online Banking bank codes. |
| 14.1 | 2019/07/11 | Pepa Simeonova | Added new money-transfer types. |
| 14.2 | 2019/07/16 | Hristo Tenev | Added API support for Business attributes. |
| 14.3 | 2019/07/22 | Pepa Simeonova | Added source_of_funds as an optional API param for OCT types (Credit, Payout) |
| 14.4 | 2019/08/01 | Stefan Petrov | Removed API for Astropay Card, Astropay Direct, Hipercard, Carulla, Emprese de Energia, Surtimax. |
| 14.5 | 2019/08/05 | Hristo Tanchev | Extended Processing and WPF APIs support for FX (Forex). |
| 14.6 | 2019/08/05 | Martin Lazarov | Added reversible amount in Preauthorization reconcile response. |
| 14.7 | 2019/08/06 | Rumen Milushev | Added MOTO flag to WPF transaction types: Authorize, Authorize3D, Sale, Sale3D. |
| 14.8 | 2019/08/07 | Yasen Angelov | Added the new optional param reminder_language to the WPF API. |
| 14.9 | 2019/08/13 | Stefan Petrov | Rebranding African Mobile payments. |
| 15.0 | 2019/08/21 | Nikolay Petrov | Added new status 'represented' in notifications for Processing and WPF APIs. |
| 15.1 | 2019/08/22 | Stefan Petrov | Added beneficiary params to refund transaction. |
| 15.2 | 2019/09/03 | Stefan Petrov | Use sync workflow for Neosurf transaction. |
| 15.3 | 2019/09/13 | Hristo Tanchev | Extended transaction response with transaction ID form card schemes. |
| 15.4 | 2019/09/13 | Yordan Pulov | Added Genesis SCA Checker service API. |
| 15.5 | 2019/09/16 | Nikolay Petrov | Added support for Account Verification to the WPF API. |
| 15.6 | 2019/09/17 | Ventsislav Dimitrov | Extended the supported Merchant Category Codes for Visa Preauthorization |
| 15.7 | 2019/09/18 | Aleksandar Krastev | Added Processed Transaction API for Card Present and Card Not Present transactions. |
| 15.8 | 2019/09/20 | Yordan Pulov | Extended Genesis SCA Checker service API documentation. |
| 15.9 | 2019/09/20 | Yordan Pulov | Added new fields and changed the endpoint for sending TransferToAPI requests. |
| 16.0 | 2019/09/24 | Rumen Milushev | Added 'agency_name' to Hotel Rental Travel attributes. |
| 16.1 | 2019/09/25 | Aleksandar Krastev | Changed filter flags of Processed Transaction API by_date and by_post_date endpoints. |
| 16.2 | 2019/10/09 | Aleksandar Krastev | Extended Chargeback API to return more attributes; Added flags filtering by origin and type of processing. |
| 16.3 | 2019/10/10 | Pepa Simeonova | Added new MPI parameters and SCA parameters. |
| 16.4 | 2019/10/16 | Aleksandar Krastev | Extended Retrieval Request API to return more attributes; Added flags filtering by origin and type of processing. |
| 16.5 | 2019/10/17 | Martin Lazarov | Extended the supported Merchant Category Codes for Visa Preauthorization. |
| 16.6 | 2019/10/21 | Yordan Pulov | Added support for UATP Travel. |
| 16.7 | 2019/10/24 | Rumen Milushev | Transferred Hotel Rentals, Car Rentals and Cruise Lines from Travel Level 3 data to Business Attributes. |
| 16.8 | 2019/10/24 | Martin Lazarov | Added Reference Transaction Unique ID to notification parameters for the reference-based transactions. |
| 16.9 | 2019/10/31 | Martin Lazarov | Added Capture tolerance for Mastercard and Maestro Preauthorizations. |
| 17.0 | 2019/11/05 | Yordan Pulov | Added new fields for TransferTo Payout. |
| 17.1 | 2019/11/05 | Nikolay Petrov | Added new supported currencies and banks for Bank Payout. |
| 17.2 | 2019/11/15 | Ventsislav Dimitrov | Added API support for purchasing Mastercard and Maestro crypto-currencies. |
| 17.3 | 2019/11/15 | Nikolay Petrov | Added optional bank params to the refund transaction. |
| 17.4 | 2019/11/19 | Stefan Petrov | Added Processed Batches API. |
| 17.5 | 2019/11/20 | Pepa Simeonova | Added sub_merchant_id as an optional dynamic descriptor param. |
| 17.6 | 2019/11/20 | Rumen Milushev | Added optional time extensions to the Processing Reconcile and WPF Reconcile APIs. |
| 17.7 | 2019/11/21 | Nikolay Petrov | Added support for Online Banking Unified Payment Interface (UPI) payment type. |

| Version | Date | Name | Description |
|---|---|---|---|
| 17.8 | 2019/11/27 | Stefan Petrov | Added Batch and Deposit Slip Numbers to Processed Transaction API response. |
| 17.9 | 2019/11/28 | Stefan Petrov | Removed API for Entercash and Banamex. |
| 18.0 | 2019/12/03 | Nikolay Petrov | Made BIC param optional for SddSale and SddInitRecurringSale transactions. |
| 18.1 | 2019/12/05 | Nikolay Petrov | Added supported bank codes for Online Banking that can be used with Netbanking payment type. |
| 18.2 | 2019/12/12 | George Naydenov | Added Trustly Select Account API. Extended parameters for Trustly Sale and Bank Pay-out. |
| 18.3 | 2019/12/18 | Rumen Milushev | Added descriptions for Airlines and Travel agencies in Business Attributes. |
| 18.4 | 2019/12/20 | George Naydenov | Added birth_date as conditionally required API param for Trustly Sale and Bank Pay-out. |
| 18.5 | 2020/01/08 | Hristo Tenev | Added Funding Account API for Card Present and Card Not Present transactions. |
| 18.6 | 2020/01/21 | Sridhar Belagod | Added Trustly Register Account API. |
| 18.7 | 2020/01/29 | Sridhar Belagod | Marked birth_date as optional param for Trustly and changed description. |
| 18.8 | 2020/01/29 | Ralitsa Borisova | Added Finnish language as part of the platform internationalization |
| 18.9 | 2020/01/30 | Sridhar Belagod | Updated supported countries for Trustly Sale. |
| 19.0 | 2020/02/04 | Dmitri Lihachev | Added API for Apple Pay. |
| 19.1 | 2020/02/05 | Sridhar Belagod | Removed unique_id param from Trustly RegisterAccount API. Updated birth_date format and extended the example to include country-specific format |
| 19.2 | 2020/02/05 | Stefan Petrov | Exposed querying and reconciling by ARN for Fraud reports, Chargebacks and Transactions. |
| 19.3 | 2020/02/21 | Ralitsa Borisova | Updated description for Trustly Sale. |
| 19.4 | 2020/02/26 | Rumen Milushev | Added batch_slip_number and deposit_slip_number as optional search params for ProcessedTransactions and ProcessedBatches. |
| 19.5 | 2020/02/26 | Rumen Milushev | Added type and card_number response params to Processed Transactions API response. |
| 19.6 | 2020/02/28 | Svilen Siderov | Updated MyBank supported countries for PPRO. |
| 19.7 | 2020/03/11 | Stanislav Mihailov | Added Payment Authorizations API for Card Present and Card Not Present authorizations. |
| 19.8 | 2020/03/18 | Nikolay Petrov | Added captured flag and capturable_amount param to Reconcile API response for authorizations. |
| 19.9 | 2020/03/18 | Stefan Petrov | Updated supported countries for PaysafeCard. |
| 20.0 | 2020/03/19 | Stefan Petrov | Removed France and United Kingdom from the supported countries for Sofort. |
| 20.1 | 2020/03/18 | Teodor Nikolov | Updated amount fields descriptions. |
| 20.2 | 2020/03/23 | George Naydenov | Added list of supported clearinghouses for Trustly register account call. |
| 20.3 | 2020/03/23 | Rumen Milushev | Added merchant_number, merchant_reference_transaction and capture_method to Chargebacks API response. |
| 20.4 | 2020/03/27 | Rumen Milushev | Unified response param to 'merchant_transaction_reference' in Payment Transactions and Chargebacks API. |
| 20.5 | 2020/04/03 | Ventsislav Dimitrov | Added asynchronous 3DSv2 support for 3D transaction types |
| 20.6 | 2020/04/03 | Svilen Siderov | Added card type, card subtype and card issuing bank to the Processing and WPF Reconcile API responses |
| 20.7 | 2020/04/08 | Stanislav Mihailov | Extended Risk data APIs (TC40/SAFE, Chargebacks, Retrieval Requests) to return a list of records by ARN or by transaction unique ID if the new list mode is enabled |
| 20.8 | 2020/04/15 | Stanislav Mihailov | Updated amount field for Card Present, Card Not Present and External payment authorizations. |
| 20.9 | 2020/04/15 | Martin Lazarov | Added details about Async Refund transaction type. |
| 21.0 | 2020/04/15 | George Naydenov | Added clearing houses list of supported IBANs and account numbers for Trustly Register Account |
| 21.1 | 2020/04/20 | Ventsislav Dimitrov | Changed the Mastercard test card numbers for card enrolled and card not participating in the 3DSv1. |
| 21.2 | 2020/04/28 | Stanislav Mihailov | Made gaming flag optional for purchase of VISA cryptocurrency transactions. |
| 21.3 | 2020/04/28 | Ventsislav Dimitrov | Added additional required recurring params for asynchronous InitRecurringSale3d using 3DSv2 authentication protocol. |
| 21.4 | 2020/05/05 | Ventsislav Dimitrov | Added more advanced merchant sequence flow diagrams for 3DSv1 and 3DSv2 authentication protocols using the Processing API. |
| 21.5 | 2020/05/05 | Ralitsa Borisova | Added Norwegian, Danish and Swedish language as part of the platform internationalization. |
| 21.6 | 2020/05/11 | Svilen Siderov | Added bank account number and bank identifier code to the Processing and WPF Reconcile API responses. |
| 21.7 | 2020/05/15 | Ventsislav Dimitrov | Added more detailed request examples for **synchronous** 3DSecure Visa and MasterCard transactions through the **3DSv1** and **3DSv2** authentication protocol. |

| Version | Date | Name | Description |
|---|---|---|---|
| 21.8 | 2020/05/15 | Ventsislav Dimitrov | Marked `xid` as out of scope for **synchronous** 3DSecure transactions using the **3DSv2** authentication protocol. |
| 21.9 | 2020/05/15 | Ventsislav Dimitrov | Extended the 3DSv2 documentation, authentication flow diagrams, and marked `usage` as required param when processing asynchronous 3DSecure transaction through 3DSv2 authentication protocol. |
| 22.0 | 2020/05/19 | Martin Lazarov | Updated the documentation related to Alipay transaction type, its transaction amount limits and currencies. |
| 22.1 | 2020/06/01 | Mario Chankov | Updated the description related to POLi payment. |
| 22.2 | 2020/06/01 | Svilen Siderov | Added Alipay Register Merchant API. |
| 22.3 | 2020/06/09 | Svilen Siderov | Added auth_start_date and auth_end_date to Payment Authorizations API request. |
| 22.4 | 2020/06/24 | Martin Lazarov | Added `authorization_code` to the API and WPF notification of various transaction types. |
| 22.5 | 2020/06/25 | Martin Lazarov | Added `retrieval_reference_number` to: API/WPF notifications, API/WPF response and API/WPF reconciliation response. |
| 22.6 | 2020/07/01 | Martin Lazarov | Removed **UPI** from Online banking supported payment types. Added new transaction type UPI. |
| 22.7 | 2020/07/07 | Nikolay Petrov | Added support for **AliPay QR** payment type in the Online Banking. |
| 22.8 | 2020/07/10 | Mario Chankov | Removed **Zimpler** and **Santander Cash**. Changes to supported countries for **MyBank**, **Safetypay** and **Santander** |

# Introduction

You can get PHP SDK from https://github.com/GenesisGateway/genesis_php

**Installation:**
Get the SDK from GitHub and load it with Composer autoloader or include it in your own autoloader (The SDK uses PSR-4 spec).

```
git clone http://github.com/GenesisGateway/genesis_php genesis_php
cd genesis_php
composer install
```

You can get Node.js SDK from https://github.com/GenesisGateway/genesis.js

**Installation:**
Get the SDK from GitHub

```
git clone http://github.com/GenesisGateway/genesis.js genesis.js
cd genesis.js
npm install
```

Or install it using npm

```
npm install genesis.js
```

You can get Java SDK from http://github.com/GenesisGateway/genesis_java

**Installation:**
Get the SDK from GitHub and install Maven plugin

```
git clone http://github.com/GenesisGateway/genesis_java genesis_java
cd genesis_java
mvn clean install
```

Finally add this to your pom.xml file:

```
<dependency>
  <groupId>com.emerchantpay.gateway</groupId>
  <artifactId>genesis-java</artifactId>
  <version>1.3.1</version>
</dependency>
```

You can get cURL from https://curl.haxx.se

This document describes the usage of the payment gateway XML/JSON API.

The API allows you to trigger all supported transactions of the gateway and to retrieve information about transactions existing in the gateway. You can also retrieve chargeback information.

The payment API is synchronous (except for 3-D secure payments), it accepts HTTP POST or XML data and returns XML data. Connections are always secured via SSL both in test and live mode. Be sure to set Content-type: text/xml in your headers.

> The API cannot be accessed via HTTP GET.

> Current Java SDKs do not have the necessary root certificate installed. You need to download the CA with your browser and import it into cacerts manually.

# Audience

This document is intended for technical staff integrating the XML/JSON API in the merchant's organization.

It is required that readers have working knowledge of programming languages, XML and JSON formats and UTF8 encodings.

# Authentication

```php
<?php

use \Genesis\Config as GenesisConfig;

// Load the pre-configured ini file...
GenesisConfig::loadSettings('/path/to/config.ini');

// ...OR, optionally, you can set the credentials manually
GenesisConfig::setEndpoint('<set_your_endpoint>');
GenesisConfig::setEnvironment('<set_your_environment>');
GenesisConfig::setUsername('<enter_your_username>');
GenesisConfig::setPassword('<enter_your_password>');
GenesisConfig::setToken('<enter_your_token>');

// You can find example configuration file in root directory of the module
```

```java
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;;
import com.emerchantpay.gateway.util.Configuration;

// Create configuration
Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

configuration.setUsername("SET_YOUR_USERNAME");
configuration.setPassword("SET_YOUR_PASSWORD");
configuration.setToken("SET_YOUR_TOKEN");
```

> You can override the path to the directory holding your configuration files (by default its config in the root directory of the module) via environmental variable NODE_CONFIG_DIR.
>
> The first file to parse configuration from, is <your-config-dir>/default.json and based on the environment variable (NODE_ENV), you can specify your custom file; for example <your-config-dir>/<NODE_ENV_NAME>.json.
>
> Its good practice to prevent web/direct access to your config directory and protect the files inside
>
> Send username and password directly in url
> `curl https://username:password@staging.gate.e-comprocessing.net:443/process/TERMINAL-TOKEN`
>
> Or use -u flag
> `curl -u username:password https://staging.gate.e-comprocessing.net:443/process/TERMINAL-TOKEN`

To interact with the payment API, you need to provide login credentials using standard HTTP Basic Authentication. (credentials can be found in your Admin interface.)

To decrease network traffic and response times, we recommend that you enforce sending authentication credentials directly in the first request.

Some implementations like e.g. the Java HTTPClient automatically try to guess the best authentication scheme. This can be overridden by setting the authorization to preemptive:

> HTTPClient.getParams.setAuthenticationPreemptivetrue;

# URLs

## Server API

API methods are called with the following structure:

`https://gate.e-comprocessing.net/process/TERMINAL_TOKEN/`

Single transaction reconciles can be done via this URL:

`https://gate.e-comprocessing.net/reconcile/TERMINAL_TOKEN/`

Date range reconciles can be done via this URL:

`https://gate.e-comprocessing.net/reconcile/by_date/TERMINAL_TOKEN/`

This URL allows retrieval of a list of supported banks based on the customers country and/or currency:

`https://gate.e-comprocessing.net/retrieve_inpay_banks/TERMINAL_TOKEN/`

For the test system use the following URLs:

`https://staging.gate.e-comprocessing.net/process/TERMINAL_TOKEN/`

`https://staging.gate.e-comprocessing.net/reconcile/TERMINAL_TOKEN/`

`https://staging.gate.e-comprocessing.net/reconcile/by_date/TERMINAL_TOKEN/`

`https://staging.gate.e-comprocessing.net/retrieve_inpay_banks/TERMINAL_TOKEN/`

---

## WPF

The URL for the WPF API create method is:

`https://wpf.e-comprocessing.net/wpf`

For the test system the URL is:

`https://staging.wpf.e-comprocessing.net/wpf`

The URL for the WPF API reconcile method is:

`https://wpf.e-comprocessing.net/wpf/reconcile`

For the test system the URL is:

`https://staging.wpf.e-comprocessing.net/wpf/reconcile`

> Terminal token and login credentials can be found in your admin panel. To use our testing environment, refer to chapter Testing

Your admin panel can be found here:

`https://merchant.e-comprocessing.net/`

---

## Consumer API

Create consumer:

`https://gate.e-comprocessing.net/v1/create_consumer/`

Retrieve consumer:

`https://gate.e-comprocessing.net/v1/retrieve_consumer/`

Update consumer:

`https://gate.e-comprocessing.net/v1/update_consumer/`

Disable consumer:

`https://gate.e-comprocessing.net/v1/disable_consumer/`

Enable consumer:

`https://gate.e-comprocessing.net/v1/enable_consumer/`

Get consumer cards:

`https://gate.e-comprocessing.net/v1/get_consumer_cards/`

For the test system use the following URLs:

`https://staging.gate.e-comprocessing.net/v1/create_consumer/`

`https://staging.gate.e-comprocessing.net/v1/update_consumer/`

`https://staging.gate.e-comprocessing.net/v1/disable_consumer/`

`https://staging.gate.e-comprocessing.net/v1/enable_consumer/`

`https://staging.gate.e-comprocessing.net/v1/get_consumer_cards/`

---

## Tokenization API

Tokenize cardholder data:

`https://gate.e-comprocessing.net/v1/tokenize/`

Detokenize cardholder data:

`https://gate.e-comprocessing.net/v1/detokenize/`

Update the cardholder data (PAN cannot be updated):

`https://gate.e-comprocessing.net/v1/update_token/`

Validate if given token is valid for the merchant:

`https://gate.e-comprocessing.net/v1/validate_token/`

Delete a token:

`https://gate.e-comprocessing.net/v1/delete_token/`

Exchange masked cardholder data for token:

`https://gate.e-comprocessing.net/v1/get_card/`

For the test system use the following URLs:

`https://staging.gate.e-comprocessing.net/v1/tokenize/`

`https://staging.gate.e-comprocessing.net/v1/detokenize/`

`https://staging.gate.e-comprocessing.net/v1/update_token/`

`https://staging.gate.e-comprocessing.net/v1/validate_token/`

`https://staging.gate.e-comprocessing.net/v1/delete_token/`

`https://staging.gate.e-comprocessing.net/v1/get_card/`

> Terminal token and login credentials can be found in your admin panel. To use our testing environment, refer to chapter Testing

---

# Genesis KYC Services JSON API

**Testing environment**

Create Consumer:

`POST https://staging.gate.e-comprocessing.net/kyc_service/create_consumer`

Update Consumer:

`POST https://staging.gate.e-comprocessing.net/kyc_service/update_consumer`

Create Transaction:

`POST https://staging.gate.e-comprocessing.net/kyc_service/create_transaction`

Update Transaction:

`POST https://staging.gate.e-comprocessing.net/kyc_service/update_transaction`

Upload Document:

`POST https://staging.gate.e-comprocessing.net/kyc_service/upload_document`

Download Document:

`POST https://staging.gate.e-comprocessing.net/kyc_service/download_document`

Verify Phone:

`POST https://staging.gate.e-comprocessing.net/kyc_service/verify_phone`

Verify Identity:

`POST https://staging.gate.e-comprocessing.net/kyc_service/verify_identity`

Verify Bank Account:

`POST https://staging.gate.e-comprocessing.net/kyc_service/verify_bank_account`

Create Authentication:

`POST https://staging.gate.e-comprocessing.net/kyc_service/create_authentication`

Update Authentication:

`POST https://staging.gate.e-comprocessing.net/kyc_service/update_authentication`

**Production environment**

Genesis KYC Services JSON API URLs for production environment:

Create Consumer:

`POST https://gate.e-comprocessing.net/kyc_service/create_consumer`

Update Consumer:

`POST https://gate.e-comprocessing.net/kyc_service/update_consumer`

Create Transaction:

`POST https://gate.e-comprocessing.net/kyc_service/create_transaction`

Update Transaction:

`POST https://gate.e-comprocessing.net/kyc_service/update_transaction`

Upload Document:

`POST https://gate.e-comprocessing.net/kyc_service/upload_document`

Download Document:

`POST https://gate.e-comprocessing.net/kyc_service/download_document`

Verify Phone:

`POST https://gate.e-comprocessing.net/kyc_service/verify_phone`

Verify Identity:

`POST https://gate.e-comprocessing.net/kyc_service/verify_identity`

Verify Bank Account:

`POST https://gate.e-comprocessing.net/kyc_service/verify_bank_account`

Create Authentication:

`POST https://gate.e-comprocessing.net/kyc_service/create_authentication`

Update Authentication:

`POST https://gate.e-comprocessing.net/kyc_service/update_authentication`

---

## Transaction API

Update expiration date:

`PUT https://gate.e-comprocessing.net/v1/transaction/expiry_date/:transaction_unique_id`

For the test system use the following URLs:

`PUT https://staging.gate.e-comprocessing.net/v1/transaction/expiry_date/:transaction_unique_id`

---

# Transactions

---

## Invoking a Transaction

A transaction is invoked via HTTPS POST, parameters are passed as XML with UTF-8 encoding.

---

## Card

**RECURRING TRANSACTIONS**

A recurring transaction describes a payment where the cardholder's account is periodically charged for a repeated delivery and use of a product or service (subscription, membership fee, etc.) over time. A recurring payment consists of an initial transaction and one or several repeated transactions. The "initial" transaction contains all relevant card and cardholder data, while the subsequent repeated transaction references an identifier which is returned with the response to the initial request.

**INIT RECURRING SALE**

An InitRecurringSale transaction initializes a recurring payment and is equal to a normal SaleTransaction except that it can be referenced as "initial" transaction in a RecurringSale transaction.

Note that if an InitRecurringSale is fully refunded, the recurring series is stopped and no more RecurringSales can be performed for that recurring series.

If an InitRecurringSale is partially refunded, the recurring series can continue with more RecurringSales

> This transaction type supports Tokenization.

> This transaction type supports Level 3 travel data.

This transaction type could require business attributes.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Recurring\InitRecurringSale');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('4200000000000000')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.recurring.InitRecurringSale;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    InitRecurringSale request = new InitRecurringSale();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("4200000000000000");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.init_recurring_sale(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "4200000000000000",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "business_attributes": {
      "event_start_date": "16-08-2020",
      "event_end_date": "25-08-2020",
      "event_organizer_id": "20192375",
      "event_id": "1912"
    },
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>init_recurring_sale</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>100</amount>
    <currency>USD</currency>
    <card_holder>Travis Pastrana</card_holder>
    <card_number>4200000000000000</card_number>
    <expiration_month>12</expiration_month>
    <expiration_year>2021</expiration_year>
    <cvv>834</cvv>
    <customer_email>travis@example.com</customer_email>
    <customer_phone>+1987987987987</customer_phone>
    <business_attributes>
      <event_start_date>16-08-2020</event_start_date>
      <event_end_date>25-08-2020</event_end_date>
      <event_organizer_id>20192375</event_organizer_id>
      <event_id>1912</event_id>
    </business_attributes>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>10178</zip_code>
      <city>Los Angeles</city>
      <state>CA</state>
      <country>US</country>
    </billing_address>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **init_recurring_sale** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |

| Parameter | Required | Format | Description |
|---|---|---|---|
| moto | optional | "true" | Signifies whether a MOTO (mail order telephone order) transaction is performed. Contacttech-support@e-comprocessing.com for more details |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| card_holder | required | string(255) | Full name of customer as printed on credit card (first name and last name at least) |
| card_number | required | 13 to 16 digits | Complete cc number of customer |
| cvv | required* | 3 to 4 digits | cvv of cc, requirement is based on terminal configuration |
| expiration_month | required | MM | Expiration month as printed on credit card |
| expiration_year | required | YYYY | Expiration year as printed on credit card |
| token | optional | string(36) | See Tokenization for more details. If present, the cardholder parameters can be omitted. Cannot be set together with `remember_card` |
| remember_card | optional | "true" | See Tokenize. Tokenizes cardholder parameters. Cannot be set together with `token` |
| consumer_id | optional | string(10) | See Consumers and Tokenization. Combine with `remember_card` to tokenize or with `token` to use token |
| **credential_on_file** | required* | | See Credential On File (COF) for more details |
| initial_customer_initiated | required* | string(18) | Initial transaction initiated by customer. Required for external tokenization, and optional for gateway-based tokenization |
| subsequent_customer_initiated | required* | string(18) | Subsequent transaction initiated by customer. Required for external tokenization, and optional for gateway-based tokenization |
| merchant_unscheduled | optional | string(20) | Transaction is initiated by the merchant |
| customer_email | required* | e-mail address | Must contain valid e-mail of customer |
| customer_phone | required* | string(32) | Must contain valid phone number of customer |
| birth_date | required* | dd-mm-yyyy | Required when MCC is a Financial Services one (e.g. MCC 6012) and either card brand is Visa or Mastercard/Maestro with UK-based merchant, UK-based bin (domestic), and DEBIT card type |
| document_id | required* | sring(20) | Document ID of the consumer. See Document ID Parameter for more details. |
| fx_rate_id | optional | integer | See Get rates for FX Service. Offers the option to use a specific FX rate to convert the transaction processing amount. Used FX rate should have the same source currency as the processing currency. Contact tech-support@e-comprocessing.com for more details |
| **business_attributes** | required* | | Check business attributes section. |
| event_start_date | required* | dd-mm-yyyy | The date when event starts in format dd-mm-yyyy |
| event_end_date | required* | dd-mm-yyyy | The date when event ends in format dd-mm-yyyy |
| event_organizer_id | required* | string | |
| event_id | required* | string | |
| date_of_order | required* | dd-mm-yyyy | The date when order was placed in format dd-mm-yyyy |
| delivery_date | required* | dd-mm-yyyy | Date of the expected delivery in format dd-mm-yyyy |
| name_of_the_supplier | required* | string | |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |

| Parameter | Required | Format | Description |
|---|---|---|---|
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required* | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |
| **dynamic_descriptor_params** | optional | | |
| merchant_name | optional | string(25) | Allows to dynamically override the charge descriptor |
| merchant_city | optional | string(13) | Allows to dynamically override the merchant phone number |
| sub_merchant_id | optional | string(15) | Allows to dynamically override the sub-merchant ID. |

`required* = conditionally required`

Successful Response

```
stdClass Object
(
    [transaction_type] => init_recurring_sale
    [status] => approved
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [consumer_id] => 123456
    [token] => ee946db8-d7db-4bb7-b608-b65b153e127d
    [avs_response_code] => 5I
    [avs_response_text] => Response provided by issuer processor; Address information not verified
    [authorization_code] => 345678
    [retrieval_reference_number] => 016813015184
    [response_code] => 00
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:13.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
    [scheme_transaction_identifier] => 019091214161031
)
```

```
<payment_response content=[
    <transaction_type content=[init_recurring_sale]>
    <status content=[approved]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <consumer_id content=[123456]>
    <token content=[ee946db8-d7db-4bb7-b608-b65b153e127d]>
    <avs_response_code content=[5I]>
    <avs_response_text content=[Response provided by issuer processor; Address information not verified]>
    <authorization_code content=[345678]>
    <retrieval_reference_number content=[016813015184]>
    <response_code content=[00]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <timestamp content=[2020-07-15T12:08:13Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[true]>
    <scheme_transaction_identifier content=[019091214161031]>
]>
```

```
{
    transaction_type: "init_recurring_sale",
    status: "approved",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    consumer_id: "123456",
    token: "ee946db8-d7db-4bb7-b608-b65b153e127d",
    avs_response_code: "5I",
    avs_response_text: "Response provided by issuer processor; Address information not verified",
    authorization_code: "345678",
    retrieval_reference_number: "016813015184",
    response_code: "00",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    timestamp: "2020-07-15T12:08:13Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "true",
    scheme_transaction_identifier: "019091214161031",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>init_recurring_sale</transaction_type>
  <status>approved</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <consumer_id>123456</consumer_id>
  <token>ee946db8-d7db-4bb7-b608-b65b153e127d</token>
  <avs_response_code>5I</avs_response_code>
  <avs_response_text>Response provided by issuer processor; Address information not verified</avs_response_text>
  <authorization_code>345678</authorization_code>
  <retrieval_reference_number>016813015184</retrieval_reference_number>
  <response_code>00</response_code>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <timestamp>2020-07-15T12:08:13Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
  <scheme_transaction_identifier>019091214161031</scheme_transaction_identifier>
</payment_transaction>
```


**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| consumer_id | string(10) | Consumer unique reference. See Consumers |
| token | string(36) | Plain-text token value. See Tokenization |
| moto | string(255) | Signifies whether a MOTO (mail order telephone order) transaction is performed. Contact tech support for more details. |
| avs_response_code | string(255) | Generated by the card network on trying to match the billing address when performing the address verification. Optional, returned if config is enabled and acquirer supports it. Check AVS Status Codes for details. |
| avs_response_text | string(255) | Gives the human response text for the AVS response code above. Optional, returned if config is enabled and acquirer supports it. Check AVS Status Codes for details. |
| authorization_code | string(6) | Generated by the card network when an authorisation has occurred, used to identify that auth. Consists of 6 alphanumeric chars |
| retrieval_reference_number | string(255) | A reference number used for tracking all messages related to a given cardholder transaction returned by some acquirers. |
| response_code | string(2) | Defines the issuer result of a transaction, the status of a message or some action taken or required. See issuer response codes for details |
| recurring_advice_code | string(2) | Optional, if received in the response from the issuer |
| recurring_advice_text | string | Optional, describes the specific recurring advice code |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08- 30T17:46:11Z |

| Parameter | Type | Description |
|---|---|---|
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| partial_approval | string(4) | Optional, set to 'true' if partial approval happened. Partially approved amount is then in the amount field. CheckPartial Approvals for details |
| sent_to_acquirer | string(255) | "true" or "false" |
| dynamic_descriptor_params | section | Optional, returned only if dynamic desc params are submitted on the API. Note here that the formatted dyn desc params are returned - as they would be submitted to the schemes for settlement. |
| scheme_transaction_identifier | string(32) | Id defined by card schemes |

Error Response

```
stdClass Object
(
    [transaction_type] => init_recurring_sale
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [response_code] => 57
    [code] => 340
    [message] => billing_address[zip_code] is invalid!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:13.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => false
)
```

```
<payment_response content=[
    <transaction_type content=[init_recurring_sale]>
    <status content=[error]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <response_code content=[57]>
    <code content=[340]>
    <message content=[billing_address[zip_code] is invalid!]>
    <timestamp content=[2020-07-15T12:08:13Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[false]>
]>
```

```
{
    transaction_type: "init_recurring_sale",
    status: "error",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    response_code: "57",
    code: "340",
    message: "billing_address[zip_code] is invalid!",
    timestamp: "2020-07-15T12:08:13Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "false",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>init_recurring_sale</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <response_code>57</response_code>
  <code>340</code>
  <message>billing_address[zip_code] is invalid!</message>
  <timestamp>2020-07-15T12:08:13Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>false</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08- 30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |
| dynamic_descriptor_params | section | Optional, returned only if dynamic desc params are submitted on the API. Note here that the formatted dyn desc params are returned - as they would be submitted to the schemes for settlement. |

**RECURRING SALE**

A RecurringSale transaction is a "repeated" transaction which follows and references a Init Recurring Sale transaction.

The card and cardholder data is omitted. Note that RecurringSales can be partially or fully refunded if configuration allows it, and this will not stop the recurring series.

This transaction type supports Level 3 travel data.

Business attributes are optional, but if submitted they will override the already supplied attributes in the initial init_recurring_sale / init_recurring_sale3d transaction.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Recurring\RecurringSale');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setReferenceId('43672')
        ->setAmount('100')
        ->setCurrency('USD');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.recurring.RecurringSale;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    RecurringSale request = new RecurringSale();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setReferenceId("43672");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```node
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.recurring_sale(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "reference_id": "43672",
    "amount": "100",
    "currency": "USD",
    "business_attributes": {
      "event_start_date": "16-08-2020",
      "event_end_date": "25-08-2020",
      "event_organizer_id": "20192375",
      "event_id": "1912"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```curl
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>recurring_sale</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <reference_id>43672</reference_id>
    <amount>100</amount>
    <currency>USD</currency>
    <business_attributes>
      <event_start_date>16-08-2020</event_start_date>
      <event_end_date>25-08-2020</event_end_date>
      <event_organizer_id>20192375</event_organizer_id>
      <event_id>1912</event_id>
    </business_attributes>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **recurring_sale** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| gaming | optional | "true" | Signifies whether a gaming transaction is performed. Gaming transactions usually use MCC 7995. Contacttech-support@e-comprocessing.com for more details |
| moto | optional | "true" | Signifies whether a MOTO (mail order telephone order) transaction is performed. Contacttech-support@e-comprocessing.com for more details |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| reference_id | required | string(32) | Unique id returned by corresponding transaction |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| **business_attributes** | required* | | Check business attributes section. |
| event_start_date | required* | dd-mm-yyyy | The date when event starts in format dd-mm-yyyy |
| event_end_date | required* | dd-mm-yyyy | The date when event ends in format dd-mm-yyyy |
| event_organizer_id | required* | string | |
| event_id | required* | string | |
| date_of_order | required* | dd-mm-yyyy | The date when order was placed in format dd-mm-yyyy |
| delivery_date | required* | dd-mm-yyyy | Date of the expected delivery in format dd-mm-yyyy |
| name_of_the_supplier | required* | string | |

`required* = conditionally required`

Successful Response

```
stdClass Object
(
    [transaction_type] => recurring_sale
    [status] => approved
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [authorization_code] => 345678
    [retrieval_reference_number] => 016813015184
    [response_code] => 00
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:14.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[recurring_sale]>
    <status content=[approved]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <authorization_code content=[345678]>
    <retrieval_reference_number content=[016813015184]>
    <response_code content=[00]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <timestamp content=[2020-07-15T12:08:14Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[true]>
]>
```

```
{
    transaction_type: "recurring_sale",
    status: "approved",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    authorization_code: "345678",
    retrieval_reference_number: "016813015184",
    response_code: "00",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    timestamp: "2020-07-15T12:08:14Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "true",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>recurring_sale</transaction_type>
  <status>approved</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <authorization_code>345678</authorization_code>
  <retrieval_reference_number>016813015184</retrieval_reference_number>
  <response_code>00</response_code>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <timestamp>2020-07-15T12:08:14Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```


**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| authorization_code | string(6) | Generated by the card network when an authorisation has occurred, used to identify that auth. Consists of 6 alphanumeric chars |
| retrieval_reference_number | string(255) | A reference number used for tracking all messages related to a given cardholder transaction returned by some acquirers. |
| response_code | string(2) | Defines the issuer result of a transaction, the status of a message or some action taken or required. Seeissuer response codes for details |
| recurring_advice_code | string(2) | Optional, if received in the response from the issuer |
| recurring_advice_text | string | Optional, describes the specific recurring advice code |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08- 30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [transaction_type] => recurring_sale
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [response_code] => 57
    [code] => 340
    [message] => billing_address[zip_code] is invalid!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:14.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => false
)
```

```
<payment_response content=[
    <transaction_type content=[recurring_sale]>
    <status content=[error]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <response_code content=[57]>
    <code content=[340]>
    <message content=[billing_address[zip_code] is invalid!]>
    <timestamp content=[2020-07-15T12:08:14Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[false]>
]>
```

```
{
    transaction_type: "recurring_sale",
    status: "error",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    response_code: "57",
    code: "340",
    message: "billing_address[zip_code] is invalid!",
    timestamp: "2020-07-15T12:08:14Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "false",
}
```

```
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>recurring_sale</transaction_type>
    <status>error</status>
    <mode>live</mode>
    <transaction_id>43671</transaction_id>
    <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
    <response_code>57</response_code>
    <code>340</code>
    <message>billing_address[zip_code] is invalid!</message>
    <timestamp>2020-07-15T12:08:14Z</timestamp>
    <descriptor>Descriptor one</descriptor>
    <amount>100</amount>
    <currency>USD</currency>
    <sent_to_acquirer>false</sent_to_acquirer>
  </payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| gaming | string(255) | Signifies whether a gaming transaction is performed. Gaming transactions usually use MCC 7995, contact tech support for more details. |
| moto | string(255) | Signifies whether a MOTO (mail order telephone order) transaction is performed. Contact tech support for more details. |
| response_code | string(2) | Defines the issuer result of a transaction, the status of a message or some action taken or required. See issuer response codes for details |
| code | integer | Error code according to Error code table |

| Parameter | Type | Description |
|---|---|---|
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08- 30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |
| dynamic_descriptor_params | section | Optional, returned only if dynamic desc params are submitted on the API. Note here that the formatted dyn desc params are returned - as they would be submitted to the schemes for settlement. |

**AUTHORIZE**

With authorize transactions, you can confirm that a credit card is valid and reserve the desired amount on the card.

After settling the transaction (e.g. shipping the goods), you can then capture the amount. The customer will not be billed until the capture has taken place, but the amount is reserved and the customer's credit card limit is reduced. Authorizes will automatically be cancelled after a certain timeframe, most likely one week.

For a typical e-commerce application it is recommended to authorize the amount on incoming orders and capture it when shipping the goods. If you are selling services or non-tangible goods, you can use the sale transaction, which combines authorize and capture.

If you choose not to serve the customer, consider to void the authorize to unfreeze the amount on the client's credit card.

Authorize transactions are also available as 3dsecure transactions

This transaction type supports Tokenization.

This transaction type supports Level 3 travel data.

This transaction type supports Preauthorizations.

This transaction type could require business attributes.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Authorize');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('4200000000000000')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.AuthorizeRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    AuthorizeRequest request = new AuthorizeRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("4200000000000000");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.authorize(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "4200000000000000",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "business_attributes": {
      "event_start_date": "16-08-2020",
      "event_end_date": "25-08-2020",
      "event_organizer_id": "20192375",
      "event_id": "1912"
    },
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>authorize</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>100</amount>
    <currency>USD</currency>
    <card_holder>Travis Pastrana</card_holder>
    <card_number>4200000000000000</card_number>
    <expiration_month>12</expiration_month>
    <expiration_year>2021</expiration_year>
    <cvv>834</cvv>
    <customer_email>travis@example.com</customer_email>
    <customer_phone>+1987987987987</customer_phone>
    <business_attributes>
      <event_start_date>16-08-2020</event_start_date>
      <event_end_date>25-08-2020</event_end_date>
      <event_organizer_id>20192375</event_organizer_id>
      <event_id>1912</event_id>
    </business_attributes>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>10178</zip_code>
      <city>Los Angeles</city>
      <state>CA</state>
      <country>US</country>
    </billing_address>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **authorize** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |

| Parameter | Required | Format | Description |
|---|---|---|---|
| gaming | optional | "true" | Signifies whether a gaming transaction is performed. Gaming transactions usually use MCC 7995. Contacttech-support@e-comprocessing.com for more details |
| moto | optional | "true" | Signifies whether a MOTO (mail order telephone order) transaction is performed. Contacttech-support@e-comprocessing.com for more details |
| crypto | optional | "true" | Signifies whether a purchase of crypto-currency transaction is performed. Must be populated when purchasing crypto-currency with a VISA card and MCC is 6051. Must be populated when purchasing crypto-currency with a MASTER or INTL MAESTRO card and MCC is one of 6051, 6211. Contact tech-support@e-comprocessing.com for more details |
| preauthorization | optional | "true" | Signifies whether a preauthorization transaction is performed. Check the Preauthorizations section or contact tech support for more details. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| card_holder | required | string(255) | Full name of customer as printed on credit card (first name and last name at least) |
| card_number | required | 13 to 16 digits | Complete cc number of customer |
| cvv | required* | 3 to 4 digits | cvv of cc, requirement is based on terminal configuration |
| expiration_month | required | MM | Expiration month as printed on credit card |
| expiration_year | required | YYYY | Expiration year as printed on credit card |
| token | optional | string(36) | See Tokenization for more details. If present, the cardholder parameters can be omitted. Cannot be set together with `remember_card` |
| remember_card | optional | "true" | See Tokenize. Tokenizes cardholder parameters. Cannot be set together with `token` |
| consumer_id | optional | string(10) | See Consumers and Tokenization. Combine with `remember_card` to tokenize or with `token` to use token |
| **credential_on_file** | required* | | See Credential On File (COF) for more details |
| initial_customer_initiated | required* | string(18) | Initial transaction initiated by customer. Required for external tokenization, and optional for gateway-based tokenization |
| subsequent_customer_initiated | required* | string(18) | Subsequent transaction initiated by customer. Required for external tokenization, and optional for gateway-based tokenization |
| merchant_unscheduled | optional | string(20) | Transaction is initiated by the merchant |
| credential_on_file_transaction_identifier | optional | string(15..32) | See Credential On File (COF) for more details |
| customer_email | required* | e-mail address | Must contain valid e-mail of customer |
| customer_phone | required* | string(32) | Must contain valid phone number of customer |
| birth_date | required* | dd-mm-yyyy | Required when MCC is a Financial Services one (e.g. MCC 6012) and either card brand is Visa or Mastercard/Maestro with UK-based merchant, UK-based bin (domestic), and DEBIT card type |
| document_id | required* | sring(20) | Document ID of the consumer. See Document ID Parameter for more details. |
| fx_rate_id | optional | integer | See Get rates for FX Service. Offers the option to use a specific FX rate to convert the transaction processing amount. Used FX rate should have the same source currency as the processing currency. Contact tech-support@e-comprocessing.com for more details |
| **business_attributes** | required* | | Check business attributes section. |
| event_start_date | required* | dd-mm-yyyy | The date when event starts in format dd-mm-yyyy |
| event_end_date | required* | dd-mm-yyyy | The date when event ends in format dd-mm-yyyy |
| event_organizer_id | required* | string | |
| event_id | required* | string | |
| date_of_order | required* | dd-mm-yyyy | The date when order was placed in format dd-mm-yyyy |
| delivery_date | required* | dd-mm-yyyy | Date of the expected delivery in format dd-mm-yyyy |
| name_of_the_supplier | required* | string | |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |

| Parameter | Required | Format | Description |
|---|---|---|---|
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required* | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |
| **dynamic_descriptor_params** | optional | | |
| merchant_name | optional | string(25) | Allows to dynamically override the charge descriptor |
| merchant_city | optional | string(13) | Allows to dynamically override the merchant phone number |
| sub_merchant_id | optional | string(15) | Allows to dynamically override the sub-merchant ID. |

`required* = conditionally required`

Successful Response

```
stdClass Object
(
    [transaction_type] => authorize
    [status] => approved
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [consumer_id] => 123456
    [token] => ee946db8-d7db-4bb7-b608-b65b153e127d
    [avs_response_code] => 5I
    [avs_response_text] => Response provided by issuer processor; Address information not verified
    [authorization_code] => 345678
    [retrieval_reference_number] => 016813015184
    [response_code] => 00
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:14.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
    [scheme_transaction_identifier] => 019091214161031
)
```

```xml
<payment_response content=[
    <transaction_type content=[authorize]>
    <status content=[approved]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <consumer_id content=[123456]>
    <token content=[ee946db8-d7db-4bb7-b608-b65b153e127d]>
    <avs_response_code content=[5I]>
    <avs_response_text content=[Response provided by issuer processor; Address information not verified]>
    <authorization_code content=[345678]>
    <retrieval_reference_number content=[016813015184]>
    <response_code content=[00]>
    <timestamp content=[2020-07-15T12:08:14Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[true]>
    <scheme_transaction_identifier content=[019091214161031]>
]>
```

```json
{
    transaction_type: "authorize",
    status: "approved",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    consumer_id: "123456",
    token: "ee946db8-d7db-4bb7-b608-b65b153e127d",
    avs_response_code: "5I",
    avs_response_text: "Response provided by issuer processor; Address information not verified",
    authorization_code: "345678",
    retrieval_reference_number: "016813015184",
    response_code: "00",
    timestamp: "2020-07-15T12:08:14Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "true",
    scheme_transaction_identifier: "019091214161031",
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>authorize</transaction_type>
  <status>approved</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <consumer_id>123456</consumer_id>
  <token>ee946db8-d7db-4bb7-b608-b65b153e127d</token>
  <avs_response_code>5I</avs_response_code>
  <avs_response_text>Response provided by issuer processor; Address information not verified</avs_response_text>
  <authorization_code>345678</authorization_code>
  <retrieval_reference_number>016813015184</retrieval_reference_number>
  <response_code>00</response_code>
  <timestamp>2020-07-15T12:08:14Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
  <scheme_transaction_identifier>019091214161031</scheme_transaction_identifier>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| consumer_id | string(10) | Consumer unique reference. See Consumers |
| token | string(36) | Plain-text token value. See Tokenization |
| gaming | string(255) | Signifies whether a gaming transaction is performed. Gaming transactions usually use MCC 7995, contact tech support for more details. |
| moto | string(255) | Signifies whether a MOTO (mail order telephone order) transaction is performed. Contact tech support for more details. |
| avs_response_code | string(255) | Generated by the card network on trying to match the billing address when performing the address verification. Optional, returned if config is enabled and acquirer supports it. Check AVS Status Codes for details. |
| avs_response_text | string(255) | Gives the human response text for the AVS response code above. Optional, returned if config is enabled and acquirer supports it. Check AVS Status Codes for details. |

| Parameter | Type | Description |
|---|---|---|
| authorization_code | string(6) | Generated by the card network when an authorisation has occurred, used to identify that auth. Consists of 6 alphanumeric chars |
| retrieval_reference_number | string(255) | A reference number used for tracking all messages related to a given cardholder transaction returned by some acquirers. |
| response_code | string(2) | Defines the issuer result of a transaction, the status of a message or some action taken or required. See issuer response codes for details |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08- 30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| partial_approval | string(4) | Optional, set to 'true' if partial approval happened. Partially approved amount is then in the amount field. Check Partial Approvals for details |
| sent_to_acquirer | string(255) | "true" or "false" |
| dynamic_descriptor_params | section | Optional, returned only if dynamic desc params are submitted on the API. Note here that the formatted dyn desc params are returned - as they would be submitted to the schemes for settlement. |
| scheme_transaction_identifier | string(32) | Id defined by card schemes |

Error Response

```
stdClass Object
(
    [transaction_type] => authorize
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 340
    [technical_message] => expiration_year is invalid
    [message] => expiration_year is invalid
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:14.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => false
)
```

```
<payment_response content=[
    <transaction_type content=[authorize]>
    <status content=[error]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <code content=[340]>
    <technical_message content=[expiration_year is invalid]>
    <message content=[expiration_year is invalid]>
    <timestamp content=[2020-07-15T12:08:14Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[false]>
]>
```

```
{
    transaction_type: "authorize",
    status: "error",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    code: "340",
    technical_message: "expiration_year is invalid",
    message: "expiration_year is invalid",
    timestamp: "2020-07-15T12:08:14Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "false",
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>authorize</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>340</code>
  <technical_message>expiration_year is invalid</technical_message>
  <message>expiration_year is invalid</message>
  <timestamp>2020-07-15T12:08:14Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>false</sent_to_acquirer>
</payment_transaction>
```


**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| gaming | string(255) | Signifies whether a gaming transaction is performed. Gaming transactions usually use MCC 7995, contact tech support for more details. |
| moto | string(255) | Signifies whether a MOTO (mail order telephone order) transaction is performed. Contact tech support for more details. |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08- 30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| dynamic_descriptor_params | section | Optional, returned only if dynamic desc params are submitted on the API. Note here that the formatted dyn desc params are returned - as they would be submitted to the schemes for settlement. |

#### CAPTURE

Capture settles a transaction which has been authorized before.

Do this when you are shipping goods, for example. A capture can only be used after an authorize on the same transaction and on the same terminal.

Therefore, the reference id of the authorized transaction is mandatory.

> You can also capture a partial amount of the initially authorized amount, e.g. if you want to give customers a discount. However, you cannot capture a higher amount than initially authorized.

> This transaction type supports Level 3 travel data.

> This transaction can be used to capture a Preauthorization.

> Business attributes are optional, but if submitted they will override the already supplied attributes in the initial authorize / authorize3d transaction.

Transaction workflow:

1. The merchant sends authorize transaction to the gateway.
2. The gateway replies to it. One of returned values is the unique id of the transaction.
3. The merchant sends capture transaction. Its reference id is unique id of authorize response.

   Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Capture');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setReferenceId('43672')
        ->setAmount('100')
        ->setCurrency('USD');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.CaptureRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    CaptureRequest request = new CaptureRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setReferenceId("43672");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.capture(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "reference_id": "43672",
    "amount": "100",
    "currency": "USD",
    "business_attributes": {
      "event_start_date": "16-08-2020",
      "event_end_date": "25-08-2020",
      "event_organizer_id": "20192375",
      "event_id": "1912"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>capture</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <reference_id>43672</reference_id>
    <amount>100</amount>
    <currency>USD</currency>
    <business_attributes>
      <event_start_date>16-08-2020</event_start_date>
      <event_end_date>25-08-2020</event_end_date>
      <event_organizer_id>20192375</event_organizer_id>
      <event_id>1912</event_id>
    </business_attributes>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **capture** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| reference_id | required | string(32) | Unique id returned by corresponding transaction |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| **business_attributes** | required* | | Check business attributes section. |
| event_start_date | required* | dd-mm-yyyy | The date when event starts in format dd-mm-yyyy |
| event_end_date | required* | dd-mm-yyyy | The date when event ends in format dd-mm-yyyy |
| event_organizer_id | required* | string | |
| event_id | required* | string | |
| date_of_order | required* | dd-mm-yyyy | The date when order was placed in format dd-mm-yyyy |
| delivery_date | required* | dd-mm-yyyy | Date of the expected delivery in format dd-mm-yyyy |
| name_of_the_supplier | required* | string | |

`required* = conditionally required`

Successful Response

```
stdClass Object
(
    [transaction_type] => capture
    [status] => approved
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [authorization_code] => 345678
    [response_code] => 00
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:14.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[capture]>
    <status content=[approved]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <authorization_code content=[345678]>
    <response_code content=[00]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <timestamp content=[2020-07-15T12:08:14Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[true]>
]>
```

```
{
    transaction_type: "capture",
    status: "approved",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    authorization_code: "345678",
    response_code: "00",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    timestamp: "2020-07-15T12:08:14Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "true",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>capture</transaction_type>
  <status>approved</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <authorization_code>345678</authorization_code>
  <response_code>00</response_code>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <timestamp>2020-07-15T12:08:14Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| authorization_code | string(6) | Generated by the card network when an authorisation has occurred, used to identify that auth. Consists of 6 alphanumeric chars |

| Parameter | Type | Description |
|---|---|---|
| response_code | string(2) | Defines the issuer result of a transaction, the status of a message or some action taken or required. Seeissuer response codes for details |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08- 30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [transaction_type] => capture
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 430
    [technical_message] => Reference transaction has already been captured, and acquirer does not support partial/multiple capture
    [message] => Transaction declined.
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:14.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => false
)
```

```
<payment_response content=[
    <transaction_type content=[capture]>
    <status content=[error]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <code content=[430]>
    <technical_message content=[Reference transaction has already been captured, and acquirer does not support partial/multiple capture]>
    <message content=[Transaction declined.]>
    <timestamp content=[2020-07-15T12:08:14Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[false]>
]>
```

```
{
    transaction_type: "capture",
    status: "error",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    code: "430",
    technical_message: "Reference transaction has already been captured, and acquirer does not support partial/multiple capture",
    message: "Transaction declined.",
    timestamp: "2020-07-15T12:08:14Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "false",
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>capture</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>430</code>
  <technical_message>Reference transaction has already been captured, and acquirer does not support partial/multiple capture</technical_message>
  <message>Transaction declined.</message>
  <timestamp>2020-07-15T12:08:14Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>false</sent_to_acquirer>
</payment_transaction>
```


**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

**SALE**

Sale transactions combine authorize and capture into one step.

Using a sale transaction, the amount is immediately billed to the customer's credit card. It can be reversed via avoid transaction on the same day of the transaction. Use sale transactions, if you are e.g. selling non-tangible goods or services.

Sale transactions are also available as 3dsecure transactions

This transaction type supports Tokenization.

This transaction type supports Level 3 travel data.

This transaction type could require business attributes.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Sale');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('4200000000000000')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.SaleRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    SaleRequest request = new SaleRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("4200000000000000");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.sale(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "4200000000000000",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "business_attributes": {
      "event_start_date": "16-08-2020",
      "event_end_date": "25-08-2020",
      "event_organizer_id": "20192375",
      "event_id": "1912"
    },
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>sale</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>100</amount>
    <currency>USD</currency>
    <card_holder>Travis Pastrana</card_holder>
    <card_number>4200000000000000</card_number>
    <expiration_month>12</expiration_month>
    <expiration_year>2021</expiration_year>
    <cvv>834</cvv>
    <customer_email>travis@example.com</customer_email>
    <customer_phone>+1987987987987</customer_phone>
    <business_attributes>
      <event_start_date>16-08-2020</event_start_date>
      <event_end_date>25-08-2020</event_end_date>
      <event_organizer_id>20192375</event_organizer_id>
      <event_id>1912</event_id>
    </business_attributes>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>10178</zip_code>
      <city>Los Angeles</city>
      <state>CA</state>
      <country>US</country>
    </billing_address>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **sale** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |

| Parameter | Required | Format | Description |
|---|---|---|---|
| gaming | optional | "true" | Signifies whether a gaming transaction is performed. Gaming transactions usually use MCC 7995. Contacttech-support@e-comprocessing.com for more details |
| moto | optional | "true" | Signifies whether a MOTO (mail order telephone order) transaction is performed. Contacttech-support@e-comprocessing.com for more details |
| crypto | optional | "true" | Signifies whether a purchase of crypto-currency transaction is performed. Must be populated when purchasing crypto-currency with a VISA card and MCC is 6051. Must be populated when purchasing crypto-currency with a MASTER or INTL MAESTRO card and MCC is one of 6051, 6211. Contact tech-support@e-comprocessing.com for more details |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| card_holder | required | string(255) | Full name of customer as printed on credit card (first name and last name at least) |
| card_number | required | 13 to 16 digits | Complete cc number of customer |
| cvv | required* | 3 to 4 digits | cvv of cc, requirement is based on terminal configuration |
| expiration_month | required | MM | Expiration month as printed on credit card |
| expiration_year | required | YYYY | Expiration year as printed on credit card |
| token | optional | string(36) | See Tokenization for more details. If present, the cardholder parameters can be omitted. Cannot be set together with `remember_card` |
| remember_card | optional | "true" | See Tokenize. Tokenizes cardholder parameters. Cannot be set together with `token` |
| consumer_id | optional | string(10) | See Consumers and Tokenization. Combine with `remember_card` to tokenize or with `token` to use token |
| **credential_on_file** | required* | | See Credential On File (COF) for more details |
| initial_customer_initiated | required* | string(18) | Initial transaction initiated by customer. Required for external tokenization, and optional for gateway-based tokenization |
| subsequent_customer_initiated | required* | string(18) | Subsequent transaction initiated by customer. Required for external tokenization, and optional for gateway-based tokenization |
| merchant_unscheduled | optional | string(20) | Transaction is initiated by the merchant |
| credential_on_file_transaction_identifier | optional | string(15..32) | See Credential On File (COF) for more details |
| customer_email | required* | e-mail address | Must contain valid e-mail of customer |
| customer_phone | required* | string(32) | Must contain valid phone number of customer |
| birth_date | required* | dd-mm-yyyy | Required when MCC is a Financial Services one (e.g. MCC 6012) and either card brand is Visa or Mastercard/Maestro with UK-based merchant, UK-based bin (domestic), and DEBIT card type |
| document_id | required* | sring(20) | Document ID of the consumer. See Document ID Parameter for more details. |
| fx_rate_id | optional | integer | See Get rates for FX Service. Offers the option to use a specific FX rate to convert the transaction processing amount. Used FX rate should have the same source currency as the processing currency. Contact tech-support@e-comprocessing.com for more details |
| **business_attributes** | required* | | Check business attributes section. |
| event_start_date | required* | dd-mm-yyyy | The date when event starts in format dd-mm-yyyy |
| event_end_date | required* | dd-mm-yyyy | The date when event ends in format dd-mm-yyyy |
| event_organizer_id | required* | string | |
| event_id | required* | string | |
| date_of_order | required* | dd-mm-yyyy | The date when order was placed in format dd-mm-yyyy |
| delivery_date | required* | dd-mm-yyyy | Date of the expected delivery in format dd-mm-yyyy |
| name_of_the_supplier | required* | string | |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |

| Parameter | Required | Format | Description |
|---|---|---|---|
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required* | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |
| **dynamic_descriptor_params** | optional | | |
| merchant_name | optional | string(25) | Allows to dynamically override the charge descriptor |
| merchant_city | optional | string(13) | Allows to dynamically override the merchant phone number |
| sub_merchant_id | optional | string(15) | Allows to dynamically override the sub-merchant ID. |

`required* = conditionally required`

Successful Response

```
stdClass Object
(
    [transaction_type] => sale
    [status] => approved
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [consumer_id] => 123456
    [token] => ee946db8-d7db-4bb7-b608-b65b153e127d
    [avs_response_code] => 5I
    [avs_response_text] => Response provided by issuer processor; Address information not verified
    [authorization_code] => 345678
    [retrieval_reference_number] => 016813015184
    [response_code] => 00
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:14.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
    [scheme_transaction_identifier] => 019091214161031
)
```

```
<payment_response content=[
    <transaction_type content=[sale]>
    <status content=[approved]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <consumer_id content=[123456]>
    <token content=[ee946db8-d7db-4bb7-b608-b65b153e127d]>
    <avs_response_code content=[5I]>
    <avs_response_text content=[Response provided by issuer processor; Address information not verified]>
    <authorization_code content=[345678]>
    <retrieval_reference_number content=[016813015184]>
    <response_code content=[00]>
    <timestamp content=[2020-07-15T12:08:14Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[true]>
    <scheme_transaction_identifier content=[019091214161031]>
]>
```

```
{
    transaction_type: "sale",
    status: "approved",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    consumer_id: "123456",
    token: "ee946db8-d7db-4bb7-b608-b65b153e127d",
    avs_response_code: "5I",
    avs_response_text: "Response provided by issuer processor; Address information not verified",
    authorization_code: "345678",
    retrieval_reference_number: "016813015184",
    response_code: "00",
    timestamp: "2020-07-15T12:08:14Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "true",
    scheme_transaction_identifier: "019091214161031",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>sale</transaction_type>
  <status>approved</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <consumer_id>123456</consumer_id>
  <token>ee946db8-d7db-4bb7-b608-b65b153e127d</token>
  <avs_response_code>5I</avs_response_code>
  <avs_response_text>Response provided by issuer processor; Address information not verified</avs_response_text>
  <authorization_code>345678</authorization_code>
  <retrieval_reference_number>016813015184</retrieval_reference_number>
  <response_code>00</response_code>
  <timestamp>2020-07-15T12:08:14Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
  <scheme_transaction_identifier>019091214161031</scheme_transaction_identifier>
</payment_transaction>
```


**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| consumer_id | string(10) | Consumer unique reference. See Consumers |
| token | string(36) | Plain-text token value. See Tokenization |
| gaming | string(255) | Signifies whether a gaming transaction is performed. Gaming transactions usually use MCC 7995, contact tech support for more details. |
| moto | string(255) | Signifies whether a MOTO (mail order telephone order) transaction is performed. Contact tech support for more details. |
| avs_response_code | string(255) | Generated by the card network on trying to match the billing address when performing the address verification. Optional, returned if config is enabled and acquirer supports it. Check AVS Status Codes for details. |
| avs_response_text | string(255) | Gives the human response text for the AVS response code above. Optional, returned if config is enabled and acquirer supports it. Check AVS Status Codes for details. |
| authorization_code | string(6) | Generated by the card network when an authorisation has occurred, used to identify that auth. Consists of 6 alphanumeric chars |
| retrieval_reference_number | string(255) | A reference number used for tracking all messages related to a given cardholder transaction returned by some acquirers. |
| response_code | string(2) | Defines the issuer result of a transaction, the status of a message or some action taken or required. See issuer response codes for details |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08- 30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| partial_approval | string(4) | Optional, set to 'true' if partial approval happened. Partially approved amount is then in the amount field. Check Partial Approvals for details |

| Parameter | Type | Description |
|---|---|---|
| sent_to_acquirer | string(255) | "true" or "false" |
| dynamic_descriptor_params | section | Optional, returned only if dynamic desc params are submitted on the API. Note here that the formatted dyn desc params are returned - as they would be submitted to the schemes for settlement. |
| scheme_transaction_identifier | string(32) | Id defined by card schemes |

Error Response

```
stdClass Object
(
    [transaction_type] => sale
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [response_code] => 57
    [code] => 340
    [technical_message] => billing_address[zip_code] is invalid!
    [message] => billing_address[zip_code] is invalid!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:14.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => false
)
```

```
<payment_response content=[
    <transaction_type content=[sale]>
    <status content=[error]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <response_code content=[57]>
    <code content=[340]>
    <technical_message content=[billing_address[zip_code] is invalid!]>
    <message content=[billing_address[zip_code] is invalid!]>
    <timestamp content=[2020-07-15T12:08:14Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[false]>
]>
```

```
{
    transaction_type: "sale",
    status: "error",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    response_code: "57",
    code: "340",
    technical_message: "billing_address[zip_code] is invalid!",
    message: "billing_address[zip_code] is invalid!",
    timestamp: "2020-07-15T12:08:14Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "false",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>sale</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <response_code>57</response_code>
  <code>340</code>
  <technical_message>billing_address[zip_code] is invalid!</technical_message>
  <message>billing_address[zip_code] is invalid!</message>
  <timestamp>2020-07-15T12:08:14Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>false</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| gaming | string(255) | Signifies whether a gaming transaction is performed. Gaming transactions usually use MCC 7995, contact tech support for more details. |
| moto | string(255) | Signifies whether a MOTO (mail order telephone order) transaction is performed. Contact tech support for more details. |
| response_code | string(2) | Defines the issuer result of a transaction, the status of a message or some action taken or required. Seeissuer response codes for details |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08- 30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |
| dynamic_descriptor_params | section | Optional, returned only if dynamic desc params are submitted on the API. Note here that the formatted dyn desc params are returned - as they would be submitted to the schemes for settlement. |

**ARGENCARD**

Argencard is a debit or credit card used in Argentina. It allows online shoppers to pay offline for their online purchases at over 150,000 physical outlets.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Argencard');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setConsumerReference('barney_rumble')
        ->setNationalId('8812128812')
        ->setBirthDate('30-12-1992')
        ->setCustomerEmail('barney@example.com')


        // Billing Address
        ->setBillingFirstName('Barney')
        ->setBillingLastName('Rubble')
        ->setBillingAddress1('14, Nerazdelni str')
        ->setBillingZipCode('1407')
        ->setBillingCity('Buenos Aires')
        ->setBillingCountry('AR')


        // Risk Params
        ->setRiskUserId('123456');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>argencard</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <return_success_url>http://www.example.com/success</return_success_url>
    <return_failure_url>http://www.example.com/failure</return_failure_url>
    <amount>100</amount>
    <currency>USD</currency>
    <consumer_reference>barney_rumble</consumer_reference>
    <national_id>8812128812</national_id>
    <birth_date>30-12-1992</birth_date>
    <customer_email>barney@example.com</customer_email>
    <billing_address>
      <first_name>Barney</first_name>
      <last_name>Rubble</last_name>
      <address1>14, Nerazdelni str</address1>
      <zip_code>1407</zip_code>
      <city>Buenos Aires</city>
      <country>AR</country>
    </billing_address>
    <risk_params>
      <user_id>123456</user_id>
    </risk_params>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **argencard** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| consumer_reference | required | string(20) | Consumer reference is a unique consumer identifier |
| national_id | required | string(20) | National Identifier number of the customer |
| birth_date | optional | dd-mm-yyyy | Required for Visa only when MCC is a Financial Services one (e.g. MCC 6012) |
| customer_email | required | e-mail address | Must contain valid e-mail of customer |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |

| Parameter | Required | Format | Description |
|---|---|---|---|
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

**Supported countries:**

| Country |
|---|
| AR |

Successful Response

```
stdClass Object
(
    [transaction_type] => argencard
    [status] => pending_async
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:14.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>argencard</transaction_type>
  <status>pending_async</status>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <timestamp>2020-07-15T12:08:14Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |

| Parameter | Type | Description |
|---|---|---|
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [transaction_type] => argencard
    [status] => error
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 110
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:14.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>argencard</transaction_type>
  <status>error</status>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>110</code>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:14Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

**AURA**

Aura is a local Brazilian credit card.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Aura');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setConsumerReference('barney_rumble')
        ->setNationalId('8812128812')
        ->setBirthDate('30-12-1992')
        ->setCustomerEmail('barney@example.com')


        // Billing Address
        ->setBillingFirstName('Barney')
        ->setBillingLastName('Rubble')
        ->setBillingAddress1('14, Nerazdelni str')
        ->setBillingZipCode('1407')
        ->setBillingCity('Salvador')
        ->setBillingCountry('BR')


        // Risk Params
        ->setRiskUserId('123456');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```curl
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>aura</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <return_success_url>http://www.example.com/success</return_success_url>
    <return_failure_url>http://www.example.com/failure</return_failure_url>
    <amount>100</amount>
    <currency>USD</currency>
    <consumer_reference>barney_rumble</consumer_reference>
    <national_id>8812128812</national_id>
    <birth_date>30-12-1992</birth_date>
    <customer_email>barney@example.com</customer_email>
    <billing_address>
      <first_name>Barney</first_name>
      <last_name>Rubble</last_name>
      <address1>14, Nerazdelni str</address1>
      <zip_code>1407</zip_code>
      <city>Salvador</city>
      <country>BR</country>
    </billing_address>
    <risk_params>
      <user_id>123456</user_id>
    </risk_params>
  </payment_transaction>'
```

### Request Parameters

| Parameter | Required | Format | Description |
|---|---|---|---|

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **aura** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| consumer_reference | required | string(20) | Consumer reference is a unique consumer identifier |
| national_id | required | string(20) | National Identifier number of the customer |
| birth_date | optional | dd-mm-yyyy | Required for Visa only when MCC is a Financial Services one (e.g. MCC 6012) |
| customer_email | required | e-mail address | Must contain valid e-mail of customer |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

**Supported countries:**

| Country |
|---|
| BR |

Successful Response

```
stdClass Object
(
    [transaction_type] => aura
    [status] => pending_async
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:14.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>aura</transaction_type>
  <status>pending_async</status>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <timestamp>2020-07-15T12:08:14Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [transaction_type] => aura
    [status] => error
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 110
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:14.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>aura</transaction_type>
  <status>error</status>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>110</code>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:14Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

**BANCONTACT**

Bancontact is a local Belgian debit card scheme. All Belgian debit cards are co-branded Bancontact and Maestro.

Transaction flow for a consumer is identical to a Maestro payment.

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>bcmc</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <return_success_url>http://www.example.com/success</return_success_url>
    <return_failure_url>http://www.example.com/failure</return_failure_url>
    <amount>100</amount>
    <currency>EUR</currency>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>12114</zip_code>
      <city>Brussels</city>
      <country>BE</country>
    </billing_address>
  </payment_transaction>'
```

Request

```
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Alternatives\PPRO');
    $request = $genesis->request();

    $request
        ->setPaymentType('bcmc')
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setAmount('100')
        ->setCurrency('EUR')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('12114')
        ->setBillingCity('Brussels')
        ->setBillingCountry('BE');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.apm.PProRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;
import java.net.MalformedURLException;
import java.net.URL;

public class GenesisExample {
  public static void main() throws MalformedURLException {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    PProRequest request = new PProRequest();

    request.setPaymentType("bcmc");
    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setReturnSuccessUrl(new URL("http://www.example.com/success"));
    request.setReturnFailureUrl(new URL("http://www.example.com/failure"));
    request.setAmount(new BigDecimal(100));
    request.setCurrency("EUR");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("12114");
    request.setBillingCity("Brussels");
    request.setBillingCountry("BE");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.ppro(
  {
    "payment_type": "bcmc",
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "return_success_url": "http://www.example.com/success",
    "return_failure_url": "http://www.example.com/failure",
    "amount": "100",
    "currency": "EUR",
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "12114",
      "city": "Brussels",
      "country": "BE"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>ppro</transaction_type>
   <payment_type>bcmc</payment_type>
   <transaction_id>43671</transaction_id>
   <usage>40208 concert tickets</usage>
   <remote_ip>245.253.2.12</remote_ip>
   <return_success_url>http://www.example.com/success</return_success_url>
   <return_failure_url>http://www.example.com/failure</return_failure_url>
   <amount>100</amount>
   <currency>EUR</currency>
   <billing_address>
     <first_name>Travis</first_name>
     <last_name>Pastrana</last_name>
     <address1>Muster Str. 12</address1>
     <zip_code>12114</zip_code>
     <city>Brussels</city>
     <country>BE</country>
   </billing_address>
 </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | **ppro** or **bcmc**. Contact tech support at tech-support@e-comprocessing.com for more details. |
| payment_type | required* | bcmc | Bancontact Mr. Cash. Contact tech support for more details |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

Supported currencies and countries:

| Currency code | Country code |
| --- | --- |
| EUR | BE |

Successful Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```


```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>bcmc</transaction_type>
  <status>pending_async</status>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <transaction_id>43671</transaction_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
  <mode>live</mode>
  <timestamp>2020-07-15T12:08:14Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>EUR</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```


Successful Response


```
stdClass Object
(
    [transaction_type] => ppro
    [status] => pending_async
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [transaction_id] => 43671
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [mode] => live
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:14.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => EUR
    [sent_to_acquirer] => true
)
```



```
<payment_response content=[
    <transaction_type content=[ppro]>
    <status content=[pending_async]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <transaction_id content=[43671]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <redirect_url content=[https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61]>
    <mode content=[live]>
    <timestamp content=[2020-07-15T12:08:14Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[EUR]>
    <sent_to_acquirer content=[true]>
]>
```



```
{
    transaction_type: "ppro",
    status: "pending_async",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    transaction_id: "43671",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    redirect_url: "https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61",
    mode: "live",
    timestamp: "2020-07-15T12:08:14Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "EUR",
    sent_to_acquirer: "true",
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>ppro</transaction_type>
  <status>pending_async</status>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <transaction_id>43671</transaction_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
  <mode>live</mode>
  <timestamp>2020-07-15T12:08:14Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>EUR</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```


**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```


```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>bcmc</transaction_type>
  <status>error</status>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <transaction_id>43671</transaction_id>
  <code>110</code>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:14Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>EUR</currency>
  <sent_to_acquirer>false</sent_to_acquirer>
</payment_transaction>
```


Error Response

```
stdClass Object
(
    [transaction_type] => ppro
    [status] => error
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [transaction_id] => 43671
    [code] => 110
    [technical_message] =>
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:14.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => EUR
    [sent_to_acquirer] => false
)
```

```
<payment_response content=[
    <transaction_type content=[ppro]>
    <status content=[error]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <transaction_id content=[43671]>
    <code content=[110]>
    <technical_message content=[]>
    <message content=[Something went wrong, please contact support!]>
    <timestamp content=[2020-07-15T12:08:14Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[EUR]>
    <sent_to_acquirer content=[false]>
]>
```

```
{
    transaction_type: "ppro",
    status: "error",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    transaction_id: "43671",
    code: "110",
    technical_message: "",
    message: "Something went wrong, please contact support!",
    timestamp: "2020-07-15T12:08:14Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "EUR",
    sent_to_acquirer: "false",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>ppro</transaction_type>
  <status>error</status>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <transaction_id>43671</transaction_id>
  <code>110</code>
  <payment_transaction>technical_message</payment_transaction>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:14Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>EUR</currency>
  <sent_to_acquirer>false</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |

| Parameter | Type | Description |
|---|---|---|
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

**CABAL**

Cabal is a local debit/credit card brand in Argentina which can be used for online purchases.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Cabal');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setConsumerReference('braney_rubble')
        ->setNationalId('8812128812')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>cabal</transaction_type>
   <transaction_id>43671</transaction_id>
   <usage>40208 concert tickets</usage>
   <remote_ip>245.253.2.12</remote_ip>
   <return_success_url>http://www.example.com/success</return_success_url>
   <return_failure_url>http://www.example.com/failure</return_failure_url>
   <amount>100</amount>
   <currency>USD</currency>
   <consumer_reference>braney_rubble</consumer_reference>
   <national_id>8812128812</national_id>
   <customer_email>travis@example.com</customer_email>
   <customer_phone>+1987987987987</customer_phone>
   <billing_address>
     <first_name>Travis</first_name>
     <last_name>Pastrana</last_name>
     <address1>Muster Str. 12</address1>
     <zip_code>10178</zip_code>
     <city>Los Angeles</city>
     <state>CA</state>
     <country>US</country>
   </billing_address>
 </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **cabal** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| consumer_reference | required | string(20) | Consumer reference is a unique consumer identifier |
| national_id | required | string(20) | National Identifier number of the customer |
| birth_date | optional | dd-mm-yyyy | Required for Visa only when MCC is a Financial Services one (e.g. MCC 6012) |
| customer_email | required | e-mail address | Must contain valid e-mail of customer |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |

| Parameter | Required | Format | Description |
|---|---|---|---|
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

**Supported countries**

| Country Name | Country Code |
|---|---|
| Argentina | AR |

Successful Response

```
stdClass Object
(
    [status] => pending_async
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [transaction_id] => 43671
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61d0
    [mode] => live
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:14.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <status>pending_async</status>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <transaction_id>43671</transaction_id>
  <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61d0</redirect_url>
  <mode>live</mode>
  <timestamp>2020-07-15T12:08:14Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| status | string(255) | Status of the transaction, see states |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [status] => error
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [transaction_id] => 43671
    [code] => 110
    [message] => Something went wrong, please contact support!
    [mode] => live
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:14.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => false
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <status>error</status>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <transaction_id>43671</transaction_id>
  <code>110</code>
  <message>Something went wrong, please contact support!</message>
  <mode>live</mode>
  <timestamp>2020-07-15T12:08:14Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>false</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| status | string(255) | Status of the transaction, see states |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

**CENCOSUD**

Cencosud is a local credit card in Argentina

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Cencosud');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setConsumerReference('barney_rumble')
        ->setNationalId('8812128812')
        ->setBirthDate('30-12-1992')
        ->setCustomerEmail('barney@example.com')


        // Billing Address
        ->setBillingFirstName('Barney')
        ->setBillingLastName('Rubble')
        ->setBillingAddress1('14, Nerazdelni str')
        ->setBillingZipCode('1407')
        ->setBillingCity('Buenos Aires')
        ->setBillingCountry('AR')


        // Risk Params
        ->setRiskUserId('123456');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```curl
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>cencosud</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <return_success_url>http://www.example.com/success</return_success_url>
    <return_failure_url>http://www.example.com/failure</return_failure_url>
    <amount>100</amount>
    <currency>USD</currency>
    <consumer_reference>barney_rumble</consumer_reference>
    <national_id>8812128812</national_id>
    <birth_date>30-12-1992</birth_date>
    <customer_email>barney@example.com</customer_email>
    <billing_address>
      <first_name>Barney</first_name>
      <last_name>Rubble</last_name>
      <address1>14, Nerazdelni str</address1>
      <zip_code>1407</zip_code>
      <city>Buenos Aires</city>
      <country>AR</country>
    </billing_address>
    <risk_params>
      <user_id>123456</user_id>
    </risk_params>
  </payment_transaction>'
```

### Request Parameters

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **cencosud** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |

| Parameter | Required | Format | Description |
|---|---|---|---|
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| consumer_reference | required | string(20) | Consumer reference is a unique consumer identifier |
| national_id | required | string(20) | National Identifier number of the customer |
| birth_date | optional | dd-mm-yyyy | Required for Visa only when MCC is a Financial Services one (e.g. MCC 6012) |
| customer_email | required | e-mail address | Must contain valid e-mail of customer |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

**Supported countries:**

| Country |
|---|
| AR |

Successful Response

```
stdClass Object
(
    [transaction_type] => cencosud
    [status] => pending_async
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:14.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>cencosud</transaction_type>
  <status>pending_async</status>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <timestamp>2020-07-15T12:08:14Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [transaction_type] => cencosud
    [status] => error
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 110
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:14.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>cencosud</transaction_type>
  <status>error</status>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>110</code>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:14Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

**CREDIT (CFT)**

Credits (also known as Credit Fund Transfer a.k.a. CFT) can be done with an initial reference transaction.

This transaction type allows you to transfer funds to a previously charged card. The amount can be higher than the charged reference. Credits can only be done on former sale, sale3d, init recurring sale, init recurring sale3d, recurring sale or capture transaction. Therefore, the **reference_id** for the corresponding transaction is mandatory.

Note that Visa credits are not authorized through the schemes but batched for offline settlement on the same day, while Mastercard/Maestro credits are authorized real-time. This means that the authorization code and issuer response code are available only for non-Visa credits.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Credit');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setReferenceId('43672')
        ->setAmount('100');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.CreditRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    CreditRequest request = new CreditRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setReferenceId("43672");
    request.setAmount(new BigDecimal(100));

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.credit(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "reference_id": "43672",
    "amount": "100"
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>credit</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <reference_id>43672</reference_id>
    <amount>100</amount>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **credit** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| reference_id | required | string(32) | Unique id returned by corresponding transaction |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| source_of_funds | optional | string | Specify the source of funds with one of`credit`, `debit`, `prepaid`, `cash`, `other_debit_account`, `other_credit_account`. |

`required* = conditionally required`

Successful Response

```
stdClass Object
(
    [transaction_type] => credit
    [status] => approved
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [authorization_code] => 345678
    [retrieval_reference_number] => 016813015184
    [response_code] => 00
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:14.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
)
```

```
<payment_response content=[
    <transaction_type content=[credit]>
    <status content=[approved]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <authorization_code content=[345678]>
    <retrieval_reference_number content=[016813015184]>
    <response_code content=[00]>
    <timestamp content=[2020-07-15T12:08:14Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
]>
```

```
{
    transaction_type: "credit",
    status: "approved",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    authorization_code: "345678",
    retrieval_reference_number: "016813015184",
    response_code: "00",
    timestamp: "2020-07-15T12:08:14Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>credit</transaction_type>
  <status>approved</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <authorization_code>345678</authorization_code>
  <retrieval_reference_number>016813015184</retrieval_reference_number>
  <response_code>00</response_code>
  <timestamp>2020-07-15T12:08:14Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
</payment_transaction>
```


**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| authorization_code | string(6) | Generated by the card network when an authorisation has occurred, used to identify that auth. Consists of 6 alphanumeric chars |
| retrieval_reference_number | string(255) | A reference number used for tracking all messages related to a given cardholder transaction returned by some acquirers. |
| response_code | string(2) | Defines the issuer result of a transaction, the status of a message or some action taken or required. Seeissuer response codes for details |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08- 30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |

Error Response


```
stdClass Object
(
    [transaction_type] => credit
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [response_code] => 57
    [code] => 410
    [technical_message] => No approved reference transaction found
    [message] => No approved reference transaction found
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:14.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
)
```



```
<payment_response content=[
    <transaction_type content=[credit]>
    <status content=[error]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <response_code content=[57]>
    <code content=[410]>
    <technical_message content=[No approved reference transaction found]>
    <message content=[No approved reference transaction found]>
    <timestamp content=[2020-07-15T12:08:14Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
]>
```

```
{
    transaction_type: "credit",
    status: "error",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    response_code: "57",
    code: "410",
    technical_message: "No approved reference transaction found",
    message: "No approved reference transaction found",
    timestamp: "2020-07-15T12:08:14Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>credit</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <response_code>57</response_code>
  <code>410</code>
  <technical_message>No approved reference transaction found</technical_message>
  <message>No approved reference transaction found</message>
  <timestamp>2020-07-15T12:08:14Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
</payment_transaction>
```


**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |

**ELO**

Elo is a local Brazilian payment card.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Elo');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setConsumerReference('barney_rumble')
        ->setNationalId('8812128812')
        ->setBirthDate('30-12-1992')
        ->setCustomerEmail('barney@example.com')


        // Billing Address
        ->setBillingFirstName('Barney')
        ->setBillingLastName('Rubble')
        ->setBillingAddress1('14, Nerazdelni str')
        ->setBillingZipCode('1407')
        ->setBillingCity('Salvador')
        ->setBillingCountry('BR')


        // Risk Params
        ->setRiskUserId('123456');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```curl
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>elo</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <return_success_url>http://www.example.com/success</return_success_url>
    <return_failure_url>http://www.example.com/failure</return_failure_url>
    <amount>100</amount>
    <currency>USD</currency>
    <consumer_reference>barney_rumble</consumer_reference>
    <national_id>8812128812</national_id>
    <birth_date>30-12-1992</birth_date>
    <customer_email>barney@example.com</customer_email>
    <billing_address>
      <first_name>Barney</first_name>
      <last_name>Rubble</last_name>
      <address1>14, Nerazdelni str</address1>
      <zip_code>1407</zip_code>
      <city>Salvador</city>
      <country>BR</country>
    </billing_address>
    <risk_params>
      <user_id>123456</user_id>
    </risk_params>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **elo** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |

| Parameter | Required | Format | Description |
|---|---|---|---|
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| consumer_reference | required | string(20) | Consumer reference is a unique consumer identifier |
| national_id | required | string(20) | National Identifier number of the customer |
| birth_date | optional | dd-mm-yyyy | Required for Visa only when MCC is a Financial Services one (e.g. MCC 6012) |
| customer_email | required | e-mail address | Must contain valid e-mail of customer |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

**Supported countries:**

| Country |
|---|
| BR |

Successful Response

```
stdClass Object
(
    [transaction_type] => elo
    [status] => pending_async
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:14.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```


```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>elo</transaction_type>
  <status>pending_async</status>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <timestamp>2020-07-15T12:08:14Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```


**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [transaction_type] => elo
    [status] => error
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 110
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:14.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>elo</transaction_type>
  <status>error</status>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>110</code>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:14Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

**NARANJA**

Naranja is a local credit card issued in Argentina which can be used for purchases over the internet.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Naranja');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setConsumerReference('barney_rumble')
        ->setNationalId('8812128812')
        ->setBirthDate('30-12-1992')
        ->setCustomerEmail('barney@example.com')


        // Billing Address
        ->setBillingFirstName('Barney')
        ->setBillingLastName('Rubble')
        ->setBillingAddress1('14, Nerazdelni str')
        ->setBillingZipCode('1407')
        ->setBillingCity('Buenos Aires')
        ->setBillingCountry('AR')


        // Risk Params
        ->setRiskUserId('123456');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```curl
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>naranja</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <return_success_url>http://www.example.com/success</return_success_url>
    <return_failure_url>http://www.example.com/failure</return_failure_url>
    <amount>100</amount>
    <currency>USD</currency>
    <consumer_reference>barney_rumble</consumer_reference>
    <national_id>8812128812</national_id>
    <birth_date>30-12-1992</birth_date>
    <customer_email>barney@example.com</customer_email>
    <billing_address>
      <first_name>Barney</first_name>
      <last_name>Rubble</last_name>
      <address1>14, Nerazdelni str</address1>
      <zip_code>1407</zip_code>
      <city>Buenos Aires</city>
      <country>AR</country>
    </billing_address>
    <risk_params>
      <user_id>123456</user_id>
    </risk_params>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **naranja** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |

| Parameter | Required | Format | Description |
|---|---|---|---|
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| consumer_reference | required | string(20) | Consumer reference is a unique consumer identifier |
| national_id | required | string(20) | National Identifier number of the customer |
| birth_date | optional | dd-mm-yyyy | Required for Visa only when MCC is a Financial Services one (e.g. MCC 6012) |
| customer_email | required | e-mail address | Must contain valid e-mail of customer |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

**Supported countries:**

| Country |
|---|
| AR |

Successful Response

```
stdClass Object
(
    [transaction_type] => naranja
    [status] => pending_async
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:15.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```


```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>naranja</transaction_type>
  <status>pending_async</status>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <timestamp>2020-07-15T12:08:15Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```


**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [transaction_type] => naranja
    [status] => error
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 110
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:15.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>naranja</transaction_type>
  <status>error</status>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>110</code>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:15Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

**NATIVA**

Nativa is an Argentinian credit card provided by the National Bank of Argentina.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Nativa');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setConsumerReference('barney_rumble')
        ->setNationalId('8812128812')
        ->setBirthDate('30-12-1992')
        ->setCustomerEmail('barney@example.com')


        // Billing Address
        ->setBillingFirstName('Barney')
        ->setBillingLastName('Rubble')
        ->setBillingAddress1('14, Nerazdelni str')
        ->setBillingZipCode('1407')
        ->setBillingCity('Buenos Aires')
        ->setBillingCountry('AR')


        // Risk Params
        ->setRiskUserId('123456');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```curl
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>nativa</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <return_success_url>http://www.example.com/success</return_success_url>
    <return_failure_url>http://www.example.com/failure</return_failure_url>
    <amount>100</amount>
    <currency>USD</currency>
    <consumer_reference>barney_rumble</consumer_reference>
    <national_id>8812128812</national_id>
    <birth_date>30-12-1992</birth_date>
    <customer_email>barney@example.com</customer_email>
    <billing_address>
      <first_name>Barney</first_name>
      <last_name>Rubble</last_name>
      <address1>14, Nerazdelni str</address1>
      <zip_code>1407</zip_code>
      <city>Buenos Aires</city>
      <country>AR</country>
    </billing_address>
    <risk_params>
      <user_id>123456</user_id>
    </risk_params>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **nativa** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |

| Parameter | Required | Format | Description |
|---|---|---|---|
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| consumer_reference | required | string(20) | Consumer reference is a unique consumer identifier |
| national_id | required | string(20) | National Identifier number of the customer |
| birth_date | optional | dd-mm-yyyy | Required for Visa only when MCC is a Financial Services one (e.g. MCC 6012) |
| customer_email | required | e-mail address | Must contain valid e-mail of customer |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

**Supported countries:**

| Country |
|---|
| AR |

Successful Response

```
stdClass Object
(
    [transaction_type] => nativa
    [status] => pending_async
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:15.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```


```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>nativa</transaction_type>
  <status>pending_async</status>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <timestamp>2020-07-15T12:08:15Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```


**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [transaction_type] => nativa
    [status] => error
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 110
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:15.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>nativa</transaction_type>
  <status>error</status>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>110</code>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:15Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

**TARJETA SHOPPING**

Tarjeta Shopping is a cash payment in Argentina.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\TarjetaShopping');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setConsumerReference('barney_rubble')
        ->setNationalId('8812128812')
        ->setBirthDate('30-12-1992')
        ->setCustomerEmail('travis@example.com')


        // Billing Address
        ->setBillingFirstName('Barney')
        ->setBillingLastName('Rubble')
        ->setBillingAddress1('14, Nerazdelni str')
        ->setBillingZipCode('1407')
        ->setBillingCity('Rosario')
        ->setBillingCountry('AR')


        // Risk Params
        ->setRiskUserId('123456');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```curl
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>tarjeta_shopping</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <return_success_url>http://www.example.com/success</return_success_url>
    <return_failure_url>http://www.example.com/failure</return_failure_url>
    <amount>100</amount>
    <currency>USD</currency>
    <consumer_reference>barney_rubble</consumer_reference>
    <national_id>8812128812</national_id>
    <birth_date>30-12-1992</birth_date>
    <customer_email>travis@example.com</customer_email>
    <billing_address>
      <first_name>Barney</first_name>
      <last_name>Rubble</last_name>
      <address1>14, Nerazdelni str</address1>
      <zip_code>1407</zip_code>
      <city>Rosario</city>
      <country>AR</country>
    </billing_address>
    <risk_params>
      <user_id>123456</user_id>
    </risk_params>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **tarjeta_shopping** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |

| Parameter | Required | Format | Description |
|---|---|---|---|
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| consumer_reference | required | string(20) | Consumer reference is a unique consumer identifier |
| national_id | required | string(20) | National Identifier number of the customer |
| birth_date | optional | string(20) | Birth date of the customer |
| customer_email | required | e-mail address | Must contain valid e-mail of customer |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

**Supported countries:**

| Country |
|---|
| AR |

Successful Response

```
stdClass Object
(
    [transaction_type] => tarjeta_shopping
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:15.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>tarjeta_shopping</transaction_type>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
  <timestamp>2020-07-15T12:08:15Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [transaction_type] => tarjeta_shopping
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 110
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:15.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>tarjeta_shopping</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>110</code>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:15Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

---

# Non-financial Transactions

**ACCOUNT VERIFICATION**

Account Verification transactions are implemented using the so-called zero-value auths.

Using an account verification transaction, the account existence for a given cardholder can be verified without any financial impact.

Note the account verification can also carry on an AVS request, thus you can also get the AVS response code and text by the schemes along with it. Refer to section AVS Status Codes for more information.

This transaction type supports Tokenization.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('NonFinancial\AccountVerification');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('4200000000000000')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.nonfinancial.AccountVerificationRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    AccountVerificationRequest request = new AccountVerificationRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("4200000000000000");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.account_verification(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "card_holder": "Travis Pastrana",
    "card_number": "4200000000000000",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>account_verification</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <card_holder>Travis Pastrana</card_holder>
    <card_number>4200000000000000</card_number>
    <expiration_month>12</expiration_month>
    <expiration_year>2021</expiration_year>
    <cvv>834</cvv>
    <customer_email>travis@example.com</customer_email>
    <customer_phone>+1987987987987</customer_phone>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>10178</zip_code>
      <city>Los Angeles</city>
      <state>CA</state>
      <country>US</country>
    </billing_address>
  </payment_transaction>'
```

Example When Issuer Supports Oct For This Pan:

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('NonFinancial\AccountVerification');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setIssuerOctEnabled('true')
        ->setRemoteIp('245.253.2.12')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('4200000000000000')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.nonfinancial.AccountVerificationRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    AccountVerificationRequest request = new AccountVerificationRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setIssuerOctEnabled("true");
    request.setRemoteIp("245.253.2.12");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("4200000000000000");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.account_verification(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "issuer_oct_enabled": true,
    "remote_ip": "245.253.2.12",
    "card_holder": "Travis Pastrana",
    "card_number": "4200000000000000",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```xml
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>account_verification</transaction_type>
   <transaction_id>43671</transaction_id>
   <usage>40208 concert tickets</usage>
   <issuer_oct_enabled>true</issuer_oct_enabled>
   <remote_ip>245.253.2.12</remote_ip>
   <card_holder>Travis Pastrana</card_holder>
   <card_number>4200000000000000</card_number>
   <expiration_month>12</expiration_month>
   <expiration_year>2021</expiration_year>
   <cvv>834</cvv>
   <customer_email>travis@example.com</customer_email>
   <customer_phone>+1987987987987</customer_phone>
   <billing_address>
     <first_name>Travis</first_name>
     <last_name>Pastrana</last_name>
     <address1>Muster Str. 12</address1>
     <zip_code>10178</zip_code>
     <city>Los Angeles</city>
     <state>CA</state>
     <country>US</country>
   </billing_address>
 </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **account_verification** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| moto | optional | "true" | Signifies whether a MOTO (mail order telephone order) transaction is performed. Contacttech-support@e-comprocessing.com for more details |
| issuer_oct_enabled | optional | true | Supported only by Visa. When submitted, Visa checks if the given PAN supports OCTs at the issuer. When not submitted, it is interpreted as a normal account verification. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| card_holder | required | string(255) | Full name of customer as printed on credit card (first name and last name at least) |
| card_number | required | 13 to 16 digits | Complete cc number of customer |
| cvv | required* | 3 to 4 digits | cvv of cc, requirement is based on terminal configuration |
| expiration_month | required | MM | Expiration month as printed on credit card |
| expiration_year | required | YYYY | Expiration year as printed on credit card |
| token | optional | string(36) | See Tokenization for more details. If present, the cardholder parameters can be omitted. Cannot be set together with `remember_card` |
| remember_card | optional | "true" | See Tokenize. Tokenizes cardholder parameters. Cannot be set together with `token` |
| consumer_id | optional | string(10) | See Consumers and Tokenization. Combine with `remember_card` to tokenize or with `token` to use token |
| **credential_on_file** | required* | | See Credential On File (COF) for more details |
| initial_customer_initiated | required* | string(18) | Initial transaction initiated by customer. Required for external tokenization, and optional for gateway-based tokenization |
| subsequent_customer_initiated | required* | string(18) | Subsequent transaction initiated by customer. Required for external tokenization, and optional for gateway-based tokenization |
| merchant_unscheduled | optional | string(20) | Transaction is initiated by the merchant |
| customer_email | required* | e-mail address | Must contain valid e-mail of customer |
| customer_phone | required* | string(32) | Must contain valid phone number of customer |
| document_id | required* | sring(20) | Document ID of the consumer. See Document ID Parameter for more details. |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |

| Parameter | Required | Format | Description |
|---|---|---|---|
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required* | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

Successful Response

```
stdClass Object
(
    [transaction_type] => account_verification
    [status] => approved
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [consumer_id] => 123456
    [token] => ee946db8-d7db-4bb7-b608-b65b153e127d
    [avs_response_code] => 5I
    [avs_response_text] => Response provided by issuer processor; Address information not verified
    [authorization_code] => 345678
    [response_code] => 00
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:15.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [sent_to_acquirer] => true
    [scheme_transaction_identifier] => 019091214161031
)
```

```
<payment_response content=[
    <transaction_type content=[account_verification]>
    <status content=[approved]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <consumer_id content=[123456]>
    <token content=[ee946db8-d7db-4bb7-b608-b65b153e127d]>
    <avs_response_code content=[5I]>
    <avs_response_text content=[Response provided by issuer processor; Address information not verified]>
    <authorization_code content=[345678]>
    <response_code content=[00]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <timestamp content=[2020-07-15T12:08:15Z]>
    <descriptor content=[Descriptor one]>
    <sent_to_acquirer content=[true]>
    <scheme_transaction_identifier content=[019091214161031]>
]>
```

```
{
    transaction_type: "account_verification",
    status: "approved",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    consumer_id: "123456",
    token: "ee946db8-d7db-4bb7-b608-b65b153e127d",
    avs_response_code: "5I",
    avs_response_text: "Response provided by issuer processor; Address information not verified",
    authorization_code: "345678",
    response_code: "00",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    timestamp: "2020-07-15T12:08:15Z",
    descriptor: "Descriptor one",
    sent_to_acquirer: "true",
    scheme_transaction_identifier: "019091214161031",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>account_verification</transaction_type>
  <status>approved</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <consumer_id>123456</consumer_id>
  <token>ee946db8-d7db-4bb7-b608-b65b153e127d</token>
  <avs_response_code>5I</avs_response_code>
  <avs_response_text>Response provided by issuer processor; Address information not verified</avs_response_text>
  <authorization_code>345678</authorization_code>
  <response_code>00</response_code>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <timestamp>2020-07-15T12:08:15Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <sent_to_acquirer>true</sent_to_acquirer>
  <scheme_transaction_identifier>019091214161031</scheme_transaction_identifier>
</payment_transaction>
```


Example When Issuer Supports Oct For This Pan:

Successful Response


```
stdClass Object
(
    [transaction_type] => account_verification
    [status] => approved
    [issuer_oct_enabled] => true
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [consumer_id] => 123456
    [token] => ee946db8-d7db-4bb7-b608-b65b153e127d
    [avs_response_code] => 5I
    [avs_response_text] => Response provided by issuer processor; Address information not verified
    [authorization_code] => 345678
    [response_code] => 00
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:15.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [sent_to_acquirer] => true
    [scheme_transaction_identifier] => 019091214161031
)
```

```
<payment_response content=[
    <transaction_type content=[account_verification]>
    <status content=[approved]>
    <issuer_oct_enabled content=[true]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <consumer_id content=[123456]>
    <token content=[ee946db8-d7db-4bb7-b608-b65b153e127d]>
    <avs_response_code content=[5I]>
    <avs_response_text content=[Response provided by issuer processor; Address information not verified]>
    <authorization_code content=[345678]>
    <response_code content=[00]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <timestamp content=[2020-07-15T12:08:15Z]>
    <descriptor content=[Descriptor one]>
    <sent_to_acquirer content=[true]>
    <scheme_transaction_identifier content=[019091214161031]>
]>
```

```
{
    transaction_type: "account_verification",
    status: "approved",
    issuer_oct_enabled: "true",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    consumer_id: "123456",
    token: "ee946db8-d7db-4bb7-b608-b65b153e127d",
    avs_response_code: "5I",
    avs_response_text: "Response provided by issuer processor; Address information not verified",
    authorization_code: "345678",
    response_code: "00",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    timestamp: "2020-07-15T12:08:15Z",
    descriptor: "Descriptor one",
    sent_to_acquirer: "true",
    scheme_transaction_identifier: "019091214161031",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>account_verification</transaction_type>
  <status>approved</status>
  <issuer_oct_enabled>true</issuer_oct_enabled>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <consumer_id>123456</consumer_id>
  <token>ee946db8-d7db-4bb7-b608-b65b153e127d</token>
  <avs_response_code>5I</avs_response_code>
  <avs_response_text>Response provided by issuer processor; Address information not verified</avs_response_text>
  <authorization_code>345678</authorization_code>
  <response_code>00</response_code>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <timestamp>2020-07-15T12:08:15Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <sent_to_acquirer>true</sent_to_acquirer>
  <scheme_transaction_identifier>019091214161031</scheme_transaction_identifier>
</payment_transaction>
```


Example When Issuer Does Not Support Oct For This Pan:

Successful Response


```
stdClass Object
(
    [transaction_type] => account_verification
    [status] => declined
    [issuer_oct_enabled] => false
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [consumer_id] => 123456
    [token] => ee946db8-d7db-4bb7-b608-b65b153e127d
    [avs_response_code] => 5I
    [avs_response_text] => Response provided by issuer processor; Address information not verified
    [authorization_code] => 345678
    [response_code] => 00
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:15.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [sent_to_acquirer] => true
    [scheme_transaction_identifier] => 019091214161031
)
```

```
<payment_response content=[
    <transaction_type content=[account_verification]>
    <status content=[declined]>
    <issuer_oct_enabled content=[false]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <consumer_id content=[123456]>
    <token content=[ee946db8-d7db-4bb7-b608-b65b153e127d]>
    <avs_response_code content=[5I]>
    <avs_response_text content=[Response provided by issuer processor; Address information not verified]>
    <authorization_code content=[345678]>
    <response_code content=[00]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <timestamp content=[2020-07-15T12:08:15Z]>
    <descriptor content=[Descriptor one]>
    <sent_to_acquirer content=[true]>
    <scheme_transaction_identifier content=[019091214161031]>
]>
```

```
{
    transaction_type: "account_verification",
    status: "declined",
    issuer_oct_enabled: "false",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    consumer_id: "123456",
    token: "ee946db8-d7db-4bb7-b608-b65b153e127d",
    avs_response_code: "5I",
    avs_response_text: "Response provided by issuer processor; Address information not verified",
    authorization_code: "345678",
    response_code: "00",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    timestamp: "2020-07-15T12:08:15Z",
    descriptor: "Descriptor one",
    sent_to_acquirer: "true",
    scheme_transaction_identifier: "019091214161031",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>account_verification</transaction_type>
  <status>declined</status>
  <issuer_oct_enabled>false</issuer_oct_enabled>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <consumer_id>123456</consumer_id>
  <token>ee946db8-d7db-4bb7-b608-b65b153e127d</token>
  <avs_response_code>5I</avs_response_code>
  <avs_response_text>Response provided by issuer processor; Address information not verified</avs_response_text>
  <authorization_code>345678</authorization_code>
  <response_code>00</response_code>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <timestamp>2020-07-15T12:08:15Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <sent_to_acquirer>true</sent_to_acquirer>
  <scheme_transaction_identifier>019091214161031</scheme_transaction_identifier>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| consumer_id | string(10) | Consumer unique reference. See Consumers |
| token | string(36) | Plain-text token value. See Tokenization |
| moto | string(255) | Signifies whether a MOTO (mail order telephone order) transaction is performed. Contact tech support for more details. |
| issuer_oct_enabled | string | Present only if merchant has submitted issuer oct enabled flag in the request to check if issuer supports OCTs for the given PAN. True if the issuer supports OCTs for this PAN, false otherwise. |
| avs_response_code | string(255) | Generated by the card network on trying to match the billing address when performing the address verification. Optional, returned if config is enabled and acquirer supports it. Check AVS Status Codes for details. |

| Parameter | Type | Description |
|---|---|---|
| avs_response_text | string(255) | Gives the human response text for the AVS response code above. Optional, returned if config is enabled and acquirer supports it. Check AVS Status Codes for details. |
| authorization_code | string(6) | Generated by the card network when an authorisation has occurred, used to identify that auth. Consists of 6 alphanumeric chars |
| response_code | string(2) | Defines the issuer result of a transaction, the status of a message or some action taken or required. See issuer response codes for details |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08- 30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| sent_to_acquirer | string(255) | "true" or "false" |
| scheme_transaction_identifier | string(32) | Id defined by card schemes |

Error Response

```
stdClass Object
(
    [transaction_type] => account_verification
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 340
    [technical_message] => expiration_year is invalid
    [message] => expiration_year is invalid
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:15.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [sent_to_acquirer] => false
)
```

```
<payment_response content=[
    <transaction_type content=[account_verification]>
    <status content=[error]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <code content=[340]>
    <technical_message content=[expiration_year is invalid]>
    <message content=[expiration_year is invalid]>
    <timestamp content=[2020-07-15T12:08:15Z]>
    <descriptor content=[Descriptor one]>
    <sent_to_acquirer content=[false]>
]>
```

```
{
    transaction_type: "account_verification",
    status: "error",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    code: "340",
    technical_message: "expiration_year is invalid",
    message: "expiration_year is invalid",
    timestamp: "2020-07-15T12:08:15Z",
    descriptor: "Descriptor one",
    sent_to_acquirer: "false",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>account_verification</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>340</code>
  <technical_message>expiration_year is invalid</technical_message>
  <message>expiration_year is invalid</message>
  <timestamp>2020-07-15T12:08:15Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <sent_to_acquirer>false</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| gaming | string(255) | Signifies whether a gaming transaction is performed. Gaming transactions usually use MCC 7995, contact tech support for more details. |
| moto | string(255) | Signifies whether a MOTO (mail order telephone order) transaction is performed. Contact tech support for more details. |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08- 30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| dynamic_descriptor_params | section | Optional, returned only if dynamic desc params are submitted on the API. Note here that the formatted dyn desc params are returned - as they would be submitted to the schemes for settlement. |

---

# 3DS Card

To learn more about 3D Secure and supported authentication protocols, please, visit 3D-Secure Section.

**AUTHORIZE 3D**

Authorize3D transactions basically have the same request as standard authorize transactions.

Authorize3D transactions can be handled synchronous or asynchronous depending on the parameters passed. If mpi params is passed the workflow will be synchronous. If notification url, return success url and return failure url are passed the workflow will be asynchronous.

To settle Authorize3D transactions, normal capture transactions are used. As the 3-D secure process already took place, there is no need to do the verification again when capturing.

This transaction type supports Tokenization.

This transaction type supports Level 3 travel data.

This transaction type supports Partial Approvals.

This transaction type supports Preauthorizations.

This transaction type could require business attributes.

Visa Synchronous 3 D Sv1 Fully Authenticated Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Authorize3D');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('4711100000000000')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')


        // MPI Synchronous 3DSv1
        ->setMpiEci('05')
        ->setMpiCavv('MDAwMDAwMDAwMDAxMTA2Njk5NFg=')
        ->setMpiXid('0pv62FIrT5qQODB7DCewKgEBAQI=')
        ->setMpiProtocolVersion('1')


        // Sca Params
        ->setScaExemption('trusted_merchant')
        ->setScaVisaMerchantId('00000000');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.Authorize3DRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    Authorize3DRequest request = new Authorize3DRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("4711100000000000");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");


    // Mpi Synchronous V1
    request.setMpiParams("{"eci"=>"05", "cavv"=>"MDAwMDAwMDAwMDAxMTA2Njk5NFg=", "xid"=>"0pv62FIrT5qQODB7DCewKgEBAQI=", "protocol_version"=>"1"}");


    // Sca Params
    request.setScaExemption("trusted_merchant");
    request.setScaVisaMerchantId("00000000");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.authorize3d(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "4711100000000000",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "business_attributes": {
      "event_start_date": "16-08-2020",
      "event_end_date": "25-08-2020",
      "event_organizer_id": "20192375",
      "event_id": "1912"
    },
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    },
    "mpi_params": {
      "eci": "05",
      "cavv": "MDAwMDAwMDAwMDAxMTA2Njk5NFg=",
      "xid": "0pv62FIrT5qQODB7DCewKgEBAQI=",
      "protocol_version": "1"
    },
    "sca_params": {
      "exemption": "trusted_merchant",
      "visa_merchant_id": "00000000"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>authorize3d</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>100</amount>
    <currency>USD</currency>
    <card_holder>Travis Pastrana</card_holder>
    <card_number>4711100000000000</card_number>
    <expiration_month>12</expiration_month>
    <expiration_year>2021</expiration_year>
    <cvv>834</cvv>
    <customer_email>travis@example.com</customer_email>
    <customer_phone>+1987987987987</customer_phone>
    <business_attributes>
      <event_start_date>16-08-2020</event_start_date>
      <event_end_date>25-08-2020</event_end_date>
      <event_organizer_id>20192375</event_organizer_id>
      <event_id>1912</event_id>
    </business_attributes>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>10178</zip_code>
      <city>Los Angeles</city>
      <state>CA</state>
      <country>US</country>
    </billing_address>
    <mpi_params>
      <eci>05</eci>
      <cavv>MDAwMDAwMDAwMDAxMTA2Njk5NFg=</cavv>
      <xid>0pv62FIrT5qQODB7DCewKgEBAQI=</xid>
      <protocol_version>1</protocol_version>
    </mpi_params>
    <sca_params>
      <exemption>trusted_merchant</exemption>
      <visa_merchant_id>00000000</visa_merchant_id>
    </sca_params>
  </payment_transaction>'
```

Master Synchronous 3 D Sv1 Fully Authenticated Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Authorize3D');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('5555555555559997')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')


        // MPI Synchronous 3DSv1
        ->setMpiEci('02')
        ->setMpiCavv('jlk/+AMJkDARYwAAA2SqCUQAAAA=')
        ->setMpiXid('UVg1QThJU1FVV1hWWVQxQkxZNU8=')
        ->setMpiProtocolVersion('1');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.Authorize3DRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    Authorize3DRequest request = new Authorize3DRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("5555555555559997");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");


    // Mpi Synchronous V1
    request.setMpiParams("{"eci"=>"02", "cavv"=>"jlk/+AMJkDARYwAAA2SqCUQAAAA=", "xid"=>"UVg1QThJU1FVV1hWVQxQkxZNU8=", "protocol_version"=>"1"}");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.authorize3d(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "5555555555559997",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "business_attributes": {
      "event_start_date": "16-08-2020",
      "event_end_date": "25-08-2020",
      "event_organizer_id": "20192375",
      "event_id": "1912"
    },
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    },
    "mpi_params": {
      "eci": "02",
      "cavv": "jlk/+AMJkDARYwAAA2SqCUQAAAA=",
      "xid": "UVg1QThJU1FVV1hWWVQxQkxZNU8=",
      "protocol_version": "1"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>authorize3d</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>100</amount>
    <currency>USD</currency>
    <card_holder>Travis Pastrana</card_holder>
    <card_number>5555555555559997</card_number>
    <expiration_month>12</expiration_month>
    <expiration_year>2021</expiration_year>
    <cvv>834</cvv>
    <customer_email>travis@example.com</customer_email>
    <customer_phone>+1987987987987</customer_phone>
    <business_attributes>
      <event_start_date>16-08-2020</event_start_date>
      <event_end_date>25-08-2020</event_end_date>
      <event_organizer_id>20192375</event_organizer_id>
      <event_id>1912</event_id>
    </business_attributes>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>10178</zip_code>
      <city>Los Angeles</city>
      <state>CA</state>
      <country>US</country>
    </billing_address>
    <mpi_params>
      <eci>02</eci>
      <cavv>jlk/+AMJkDARYwAAA2SqCUQAAAA=</cavv>
      <xid>UVg1QThJU1FVV1hWWVQxQkxZNU8=</xid>
      <protocol_version>1</protocol_version>
    </mpi_params>
  </payment_transaction>'
```

Visa Synchronous 3 D Sv1 Attempted Authentication Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Authorize3D');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('4711100000000000')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')


        // MPI Synchronous 3DSv1
        ->setMpiEci('06')
        ->setMpiCavv('MDAwMDAwMDAwMDAxMTA2Njk5NFg=')
        ->setMpiXid('0pv62FIrT5qQODB7DCewKgEBAQI=')
        ->setMpiProtocolVersion('1')


        // Sca Params
        ->setScaExemption('trusted_merchant')
        ->setScaVisaMerchantId('00000000');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.Authorize3DRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    Authorize3DRequest request = new Authorize3DRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("4711100000000000");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");


    // Mpi Synchronous V1
    request.setMpiParams("{"eci"=>"06", "cavv"=>"MDAwMDAwMDAwMDAxMTA2Njk5NFg=", "xid"=>"0pv62FIrT5qQODB7DCewKgEBAQI=", "protocol_version"=>"1"}");


    // Sca Params
    request.setScaExemption("trusted_merchant");
    request.setScaVisaMerchantId("00000000");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.authorize3d(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "4711100000000000",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "business_attributes": {
      "event_start_date": "16-08-2020",
      "event_end_date": "25-08-2020",
      "event_organizer_id": "20192375",
      "event_id": "1912"
    },
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    },
    "mpi_params": {
      "eci": "06",
      "cavv": "MDAwMDAwMDAwMDAxMTA2Njk5NFg=",
      "xid": "0pv62FIrT5qQODB7DCewKgEBAQI=",
      "protocol_version": "1"
    },
    "sca_params": {
      "exemption": "trusted_merchant",
      "visa_merchant_id": "00000000"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>authorize3d</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>100</amount>
    <currency>USD</currency>
    <card_holder>Travis Pastrana</card_holder>
    <card_number>4711100000000000</card_number>
    <expiration_month>12</expiration_month>
    <expiration_year>2021</expiration_year>
    <cvv>834</cvv>
    <customer_email>travis@example.com</customer_email>
    <customer_phone>+1987987987987</customer_phone>
    <business_attributes>
      <event_start_date>16-08-2020</event_start_date>
      <event_end_date>25-08-2020</event_end_date>
      <event_organizer_id>20192375</event_organizer_id>
      <event_id>1912</event_id>
    </business_attributes>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>10178</zip_code>
      <city>Los Angeles</city>
      <state>CA</state>
      <country>US</country>
    </billing_address>
    <mpi_params>
      <eci>06</eci>
      <cavv>MDAwMDAwMDAwMDAxMTA2Njk5NFg=</cavv>
      <xid>0pv62FIrT5qQODB7DCewKgEBAQI=</xid>
      <protocol_version>1</protocol_version>
    </mpi_params>
    <sca_params>
      <exemption>trusted_merchant</exemption>
      <visa_merchant_id>00000000</visa_merchant_id>
    </sca_params>
  </payment_transaction>'
```

Master Synchronous 3 D Sv1 Attempted Authentication Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Authorize3D');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('5555555555559997')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')


        // MPI Synchronous 3DSv1
        ->setMpiEci('01')
        ->setMpiCavv('hlk/+AMJkDARYwAAA2SqCUQAAAA=')
        ->setMpiXid('UVg1QThJU1FVV1hWWVQxQkxZNU8=')
        ->setMpiProtocolVersion('1');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.Authorize3DRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    Authorize3DRequest request = new Authorize3DRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("5555555555559997");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");


    // Mpi Synchronous V1
    request.setMpiParams("{"eci"=>"01", "cavv"=>"hlk/+AMJkDARYwAAA2SqCUQAAAA=", "xid"=>"UVg1QThJU1FVV1hWVQxQkxZNU8=", "protocol_version"=>"1"}");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.authorize3d(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "5555555555559997",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "business_attributes": {
      "event_start_date": "16-08-2020",
      "event_end_date": "25-08-2020",
      "event_organizer_id": "20192375",
      "event_id": "1912"
    },
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    },
    "mpi_params": {
      "eci": "01",
      "cavv": "hlk/+AMJkDARYwAAA2SqCUQAAAA=",
      "xid": "UVg1QThJU1FVV1hWWVQxQkxZNU8=",
      "protocol_version": "1"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>authorize3d</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>100</amount>
    <currency>USD</currency>
    <card_holder>Travis Pastrana</card_holder>
    <card_number>5555555555559997</card_number>
    <expiration_month>12</expiration_month>
    <expiration_year>2021</expiration_year>
    <cvv>834</cvv>
    <customer_email>travis@example.com</customer_email>
    <customer_phone>+1987987987987</customer_phone>
    <business_attributes>
      <event_start_date>16-08-2020</event_start_date>
      <event_end_date>25-08-2020</event_end_date>
      <event_organizer_id>20192375</event_organizer_id>
      <event_id>1912</event_id>
    </business_attributes>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>10178</zip_code>
      <city>Los Angeles</city>
      <state>CA</state>
      <country>US</country>
    </billing_address>
    <mpi_params>
      <eci>01</eci>
      <cavv>hlk/+AMJkDARYwAAA2SqCUQAAAA=</cavv>
      <xid>UVg1QThJU1FVV1hWWVQxQkxZNU8=</xid>
      <protocol_version>1</protocol_version>
    </mpi_params>
  </payment_transaction>'
```

Asynchronous 3 D Sv1 Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Authorize3D');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('4200000000000000')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')
        ->setNotificationUrl('https://www.example.com/notification')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.Authorize3DRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;
import java.net.MalformedURLException;
import java.net.URL;

public class GenesisExample {
  public static void main() throws MalformedURLException {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    Authorize3DRequest request = new Authorize3DRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("4200000000000000");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");
    request.setNotificationUrl(new URL("https://www.example.com/notification"));
    request.setReturnSuccessUrl(new URL("http://www.example.com/success"));
    request.setReturnFailureUrl(new URL("http://www.example.com/failure"));

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.authorize3d(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "4200000000000000",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    },
    "notification_url": "https://www.example.com/notification",
    "return_success_url": "http://www.example.com/success",
    "return_failure_url": "http://www.example.com/failure"
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>authorize3d</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>100</amount>
    <currency>USD</currency>
    <card_holder>Travis Pastrana</card_holder>
    <card_number>4200000000000000</card_number>
    <expiration_month>12</expiration_month>
    <expiration_year>2021</expiration_year>
    <cvv>834</cvv>
    <customer_email>travis@example.com</customer_email>
    <customer_phone>+1987987987987</customer_phone>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>10178</zip_code>
      <city>Los Angeles</city>
      <state>CA</state>
      <country>US</country>
    </billing_address>
    <notification_url>https://www.example.com/notification</notification_url>
    <return_success_url>http://www.example.com/success</return_success_url>
    <return_failure_url>http://www.example.com/failure</return_failure_url>
  </payment_transaction>'
```

Visa Synchronous 3 D Sv2 Fully Authenticated Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Authorize3D');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('4012000000060085')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')


        // MPI Synchronous 3DSv2
        ->setMpiEci('05')
        ->setMpiCavv('MDAwMDAwMDAwMDAxMTA2Njk5NFg=')
        ->setMpiProtocolVersion('2')
        ->setMpiDirectoryServerId('f3b3dc50-a98a-0138-0f10-00505690f91e')


        // Sca Params
        ->setScaExemption('trusted_merchant')
        ->setScaVisaMerchantId('00000000');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.Authorize3DRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    Authorize3DRequest request = new Authorize3DRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("4012000000060085");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");


    // Mpi Synchronous V2
    request.setMpiParams("{"eci"=>"05", "cavv"=>"MDAwMDAwMDAwMDAxMTA2Njk5NFg=", "protocol_version"=>"2", "directory_server_id"=>"f3b3dc50-a98a-0138-0f10-00505690f91e"}");


    // Sca Params
    request.setScaExemption("trusted_merchant");
    request.setScaVisaMerchantId("00000000");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.authorize3d(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "4012000000060085",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "business_attributes": {
      "event_start_date": "16-08-2020",
      "event_end_date": "25-08-2020",
      "event_organizer_id": "20192375",
      "event_id": "1912"
    },
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    },
    "mpi_params": {
      "eci": "05",
      "cavv": "MDAwMDAwMDAwMDAxMTA2Njk5NFg=",
      "protocol_version": "2",
      "directory_server_id": "f3b3dc50-a98a-0138-0f10-00505690f91e"
    },
    "sca_params": {
      "exemption": "trusted_merchant",
      "visa_merchant_id": "00000000"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>authorize3d</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>100</amount>
    <currency>USD</currency>
    <card_holder>Travis Pastrana</card_holder>
    <card_number>4012000000060085</card_number>
    <expiration_month>12</expiration_month>
    <expiration_year>2021</expiration_year>
    <cvv>834</cvv>
    <customer_email>travis@example.com</customer_email>
    <customer_phone>+1987987987987</customer_phone>
    <business_attributes>
      <event_start_date>16-08-2020</event_start_date>
      <event_end_date>25-08-2020</event_end_date>
      <event_organizer_id>20192375</event_organizer_id>
      <event_id>1912</event_id>
    </business_attributes>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>10178</zip_code>
      <city>Los Angeles</city>
      <state>CA</state>
      <country>US</country>
    </billing_address>
    <mpi_params>
      <eci>05</eci>
      <cavv>MDAwMDAwMDAwMDAxMTA2Njk5NFg=</cavv>
      <protocol_version>2</protocol_version>
      <directory_server_id>f3b3dc50-a98a-0138-0f10-00505690f91e</directory_server_id>
    </mpi_params>
    <sca_params>
      <exemption>trusted_merchant</exemption>
      <visa_merchant_id>00000000</visa_merchant_id>
    </sca_params>
  </payment_transaction>'
```

Master Synchronous 3 D Sv2 Fully Authenticated Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Authorize3D');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('5555555555559997')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')


        // MPI Synchronous 3DSv2
        ->setMpiEci('02')
        ->setMpiCavv('kQOImea0C/2ta1NZl4Hhwslmomqj')
        ->setMpiProtocolVersion('2')
        ->setMpiDirectoryServerId('f3b7c040-a98a-0138-0f10-00505690f91e');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.Authorize3DRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    Authorize3DRequest request = new Authorize3DRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("5555555555559997");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");


    // Mpi Synchronous V2
    request.setMpiParams("{"eci"=>"02", "cavv"=>"kQOImea0C/2ta1NZl4Hhwslmomqj", "protocol_version"=>"2", "directory_server_id"=>"f3b7c040-a98a-0138-0f10-00505690f91e"}");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.authorize3d(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "5555555555559997",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "business_attributes": {
      "event_start_date": "16-08-2020",
      "event_end_date": "25-08-2020",
      "event_organizer_id": "20192375",
      "event_id": "1912"
    },
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    },
    "mpi_params": {
      "eci": "02",
      "cavv": "kQOImea0C/2ta1NZl4Hhwslmomqj",
      "protocol_version": "2",
      "directory_server_id": "f3b7c040-a98a-0138-0f10-00505690f91e"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>authorize3d</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>100</amount>
    <currency>USD</currency>
    <card_holder>Travis Pastrana</card_holder>
    <card_number>5555555555559997</card_number>
    <expiration_month>12</expiration_month>
    <expiration_year>2021</expiration_year>
    <cvv>834</cvv>
    <customer_email>travis@example.com</customer_email>
    <customer_phone>+1987987987987</customer_phone>
    <business_attributes>
      <event_start_date>16-08-2020</event_start_date>
      <event_end_date>25-08-2020</event_end_date>
      <event_organizer_id>20192375</event_organizer_id>
      <event_id>1912</event_id>
    </business_attributes>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>10178</zip_code>
      <city>Los Angeles</city>
      <state>CA</state>
      <country>US</country>
    </billing_address>
    <mpi_params>
      <eci>02</eci>
      <cavv>kQOImea0C/2ta1NZl4Hhwslmomqj</cavv>
      <protocol_version>2</protocol_version>
      <directory_server_id>f3b7c040-a98a-0138-0f10-00505690f91e</directory_server_id>
    </mpi_params>
  </payment_transaction>'
```

Visa Synchronous 3 D Sv2 Attempted Authentication Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Authorize3D');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('4012000000060085')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')


        // MPI Synchronous 3DSv2
        ->setMpiEci('06')
        ->setMpiCavv('MDAwMDAwMDAwMDAxMTA2Njk5NFg=')
        ->setMpiProtocolVersion('2')
        ->setMpiDirectoryServerId('f3bb5db0-a98a-0138-0f10-00505690f91e')


        // Sca Params
        ->setScaExemption('trusted_merchant')
        ->setScaVisaMerchantId('00000000');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.Authorize3DRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    Authorize3DRequest request = new Authorize3DRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("4012000000060085");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");


    // Mpi Synchronous V2
    request.setMpiParams("{"eci"=>"06", "cavv"=>"MDAwMDAwMDAwMDAxMTA2Njk5NFg=", "protocol_version"=>"2", "directory_server_id"=>"f3bb5db0-a98a-0138-0f10-00505690f91e"}");


    // Sca Params
    request.setScaExemption("trusted_merchant");
    request.setScaVisaMerchantId("00000000");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.authorize3d(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "4012000000060085",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "business_attributes": {
      "event_start_date": "16-08-2020",
      "event_end_date": "25-08-2020",
      "event_organizer_id": "20192375",
      "event_id": "1912"
    },
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    },
    "mpi_params": {
      "eci": "06",
      "cavv": "MDAwMDAwMDAwMDAxMTA2Njk5NFg=",
      "protocol_version": "2",
      "directory_server_id": "f3bb5db0-a98a-0138-0f10-00505690f91e"
    },
    "sca_params": {
      "exemption": "trusted_merchant",
      "visa_merchant_id": "00000000"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>authorize3d</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>100</amount>
    <currency>USD</currency>
    <card_holder>Travis Pastrana</card_holder>
    <card_number>4012000000060085</card_number>
    <expiration_month>12</expiration_month>
    <expiration_year>2021</expiration_year>
    <cvv>834</cvv>
    <customer_email>travis@example.com</customer_email>
    <customer_phone>+1987987987987</customer_phone>
    <business_attributes>
      <event_start_date>16-08-2020</event_start_date>
      <event_end_date>25-08-2020</event_end_date>
      <event_organizer_id>20192375</event_organizer_id>
      <event_id>1912</event_id>
    </business_attributes>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>10178</zip_code>
      <city>Los Angeles</city>
      <state>CA</state>
      <country>US</country>
    </billing_address>
    <mpi_params>
      <eci>06</eci>
      <cavv>MDAwMDAwMDAwMDAxMTA2Njk5NFg=</cavv>
      <protocol_version>2</protocol_version>
      <directory_server_id>f3bb5db0-a98a-0138-0f10-00505690f91e</directory_server_id>
    </mpi_params>
    <sca_params>
      <exemption>trusted_merchant</exemption>
      <visa_merchant_id>00000000</visa_merchant_id>
    </sca_params>
  </payment_transaction>'
```

Master Synchronous 3 D Sv2 Attempted Authentication Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Authorize3D');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('5555555555559997')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')


        // MPI Synchronous 3DSv2
        ->setMpiEci('01')
        ->setMpiCavv('kFOImea0C/2ta1NZl4Hhwslmomqj')
        ->setMpiProtocolVersion('2')
        ->setMpiDirectoryServerId('f3bef860-a98a-0138-0f10-00505690f91e');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.Authorize3DRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    Authorize3DRequest request = new Authorize3DRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("5555555555559997");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");


    // Mpi Synchronous V2
    request.setMpiParams("{"eci"=>"01", "cavv"=>"kFOImea0C/2ta1NZl4Hhwslmomqj", "protocol_version"=>"2", "directory_server_id"=>"f3bef860-a98a-0138-0f10-00505690f91e"}");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.authorize3d(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "5555555555559997",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "business_attributes": {
      "event_start_date": "16-08-2020",
      "event_end_date": "25-08-2020",
      "event_organizer_id": "20192375",
      "event_id": "1912"
    },
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    },
    "mpi_params": {
      "eci": "01",
      "cavv": "kFOImea0C/2ta1NZl4Hhwslmomqj",
      "protocol_version": "2",
      "directory_server_id": "f3bef860-a98a-0138-0f10-00505690f91e"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>authorize3d</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>100</amount>
    <currency>USD</currency>
    <card_holder>Travis Pastrana</card_holder>
    <card_number>5555555555559997</card_number>
    <expiration_month>12</expiration_month>
    <expiration_year>2021</expiration_year>
    <cvv>834</cvv>
    <customer_email>travis@example.com</customer_email>
    <customer_phone>+1987987987987</customer_phone>
    <business_attributes>
      <event_start_date>16-08-2020</event_start_date>
      <event_end_date>25-08-2020</event_end_date>
      <event_organizer_id>20192375</event_organizer_id>
      <event_id>1912</event_id>
    </business_attributes>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>10178</zip_code>
      <city>Los Angeles</city>
      <state>CA</state>
      <country>US</country>
    </billing_address>
    <mpi_params>
      <eci>01</eci>
      <cavv>kFOImea0C/2ta1NZl4Hhwslmomqj</cavv>
      <protocol_version>2</protocol_version>
      <directory_server_id>f3bef860-a98a-0138-0f10-00505690f91e</directory_server_id>
    </mpi_params>
  </payment_transaction>'
```

Asynchronous 3 D Sv2 Frictionless Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Authorize3D');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('4012000000060085')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')
        ->setNotificationUrl('https://www.example.com/notification')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')


        // Threeds V2 Params
        ->setThreedsControl('{"device_type"=>"browser", "challenge_window_size"=>"full_screen", "challenge_indicator"=>"preference"}')
        ->setThreedsPurchase('{"category"=>"service"}')
        ->setThreedsRecurring('{"expiration_date"=>"16-01-2021", "frequency"=>30}')
        ->setThreedsMerchantRisk('{"shipping_indicator"=>"verified_address", "delivery_timeframe"=>"electronic", "reorder_items_indicator"=>"reordered", "pre_order_purchase_indicator"=>"merchandise_available",
        ->setThreedsCardHolderAccount('{"creation_date"=>"16-07-2019", "update_indicator"=>"more_than_60days", "last_change_date"=>"16-04-2020", "password_change_indicator"=>"no_change", "password_change_date"
        ->setThreedsBrowser('{"accept_header"=>"Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser", "java_enabled"=>"true", "language"=>"en-GB", "color_depth"=>"
        ->setThreedsSdk('{"interface"=>"native", "ui_types"=>{"ui_type"=>"multi_select"}, "application_id"=>"fc1650c0-5778-0138-8205-2cbc32a32d65", "encrypted_data"=>"encrypted-data-here", "ephemeral_public_ke

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.Authorize3DRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;
import java.net.MalformedURLException;
import java.net.URL;

public class GenesisExample {
  public static void main() throws MalformedURLException {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    Authorize3DRequest request = new Authorize3DRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("4012000000060085");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");
    request.setNotificationUrl(new URL("https://www.example.com/notification"));
    request.setReturnSuccessUrl(new URL("http://www.example.com/success"));
    request.setReturnFailureUrl(new URL("http://www.example.com/failure"));


    // Threeds V2 Params
    request.setThreedsControl("{"device_type"=>"browser", "challenge_window_size"=>"full_screen", "challenge_indicator"=>"preference"}");
    request.setThreedsPurchase("{"category"=>"service"}");
    request.setThreedsRecurring("{"expiration_date"=>"16-01-2021", "frequency"=>30}");
    request.setThreedsMerchantRisk("{"shipping_indicator"=>"verified_address", "delivery_timeframe"=>"electronic", "reorder_items_indicator"=>"reordered", "pre_order_purchase_indicator"=>"merchandise_available
    request.setThreedsCardHolderAccount("{"creation_date"=>"16-07-2019", "update_indicator"=>"more_than_60days", "last_change_date"=>"16-04-2020", "password_change_indicator"=>"no_change", "password_change_da
    request.setThreedsBrowser("{"accept_header"=>"Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser", "java_enabled"=>"true", "language"=>"en-GB", "color_depth"
    request.setThreedsSdk("{"interface"=>"native", "ui_types"=>{"ui_type"=>"multi_select"}, "application_id"=>"fc1650c0-5778-0138-8205-2cbc32a32d65", "encrypted_data"=>"encrypted-data-here", "ephemeral_public_

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.authorize3d(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "4012000000060085",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "billing_address": {
```

```
      "billing_address": {
        "first_name": "Travis",
        "last_name": "Pastrana",
        "address1": "Muster Str. 12",
        "zip_code": "10178",
        "city": "Los Angeles",
        "state": "CA",
        "country": "US"
      },
      "notification_url": "https://www.example.com/notification",
      "return_success_url": "http://www.example.com/success",
      "return_failure_url": "http://www.example.com/failure",
      "threeds_v2_params": {
        "control": {
          "device_type": "browser",
          "challenge_window_size": "full_screen",
          "challenge_indicator": "preference"
        },
        "purchase": {
          "category": "service"
        },
        "recurring": {
          "expiration_date": "16-01-2021",
          "frequency": 30
        },
        "merchant_risk": {
          "shipping_indicator": "verified_address",
          "delivery_timeframe": "electronic",
          "reorder_items_indicator": "reordered",
          "pre_order_purchase_indicator": "merchandise_available",
          "pre_order_date": "16-08-2020",
          "gift_card": "true",
          "gift_card_count": "2"
        },
        "card_holder_account": {
          "creation_date": "16-07-2019",
          "update_indicator": "more_than_60days",
          "last_change_date": "16-04-2020",
          "password_change_indicator": "no_change",
          "password_change_date": "01-07-2020",
          "shipping_address_usage_indicator": "current_transaction",
          "shipping_address_date_first_used": "11-07-2020",
          "transactions_activity_last_24_hours": "2",
          "transactions_activity_previous_year": "10",
          "provision_attempts_last_24_hours": "1",
          "purchases_count_last_6_months": "5",
          "suspicious_activity_indicator": "no_suspicious_observed",
          "registration_indicator": "30_to_60_days",
          "registration_date": "16-07-2018"
        },
        "browser": {
          "accept_header": "Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser",
          "java_enabled": "true",
          "language": "en-GB",
          "color_depth": "32",
          "screen_height": "900",
          "screen_width": "1440",
          "time_zone_offset": "-120",
          "user_agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.97 Safari/537.36"
        },
        "sdk": {
          "interface": "native",
          "ui_types": {
            "ui_type": "multi_select"
          },
          "application_id": "fc1650c0-5778-0138-8205-2cbc32a32d65",
          "encrypted_data": "encrypted-data-here",
          "ephemeral_public_key_pair": "public-key-pair",
          "max_timeout": "10",
          "reference_number": "sdk-reference-number-here"
        }
      }
    }
  ).send()
    .then(success)
    .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>authorize3d</transaction_type>
   <transaction_id>43671</transaction_id>
   <usage>40208 concert tickets</usage>
   <remote_ip>245.253.2.12</remote_ip>
   <amount>100</amount>
   <currency>USD</currency>
   <card_holder>Travis Pastrana</card_holder>
   <card_number>4012000000060085</card_number>
   <expiration_month>12</expiration_month>
   <expiration_year>2021</expiration_year>
   <cvv>834</cvv>
   <customer_email>travis@example.com</customer_email>
   <customer_phone>+1987987987987</customer_phone>
   <billing_address>
     <first_name>Travis</first_name>
     <last_name>Pastrana</last_name>
     <address1>Muster Str. 12</address1>
     <zip_code>10178</zip_code>
     <city>Los Angeles</city>
     <state>CA</state>
     <country>US</country>
   </billing_address>
   <notification_url>https://www.example.com/notification</notification_url>
   <return_success_url>http://www.example.com/success</return_success_url>
   <return_failure_url>http://www.example.com/failure</return_failure_url>
   <threeds_v2_params>
     <control>
       <device_type>browser</device_type>
       <challenge_window_size>full_screen</challenge_window_size>
       <challenge_indicator>preference</challenge_indicator>
     </control>
     <purchase>
       <category>service</category>
     </purchase>
     <merchant_risk>
       <shipping_indicator>verified_address</shipping_indicator>
       <delivery_timeframe>electronic</delivery_timeframe>
       <reorder_items_indicator>reordered</reorder_items_indicator>
       <pre_order_purchase_indicator>merchandise_available</pre_order_purchase_indicator>
       <pre_order_date>16-08-2020</pre_order_date>
       <gift_card>true</gift_card>
       <gift_card_count>2</gift_card_count>
     </merchant_risk>
     <card_holder_account>
       <creation_date>16-07-2019</creation_date>
       <update_indicator>more_than_60days</update_indicator>
       <last_change_date>16-04-2020</last_change_date>
       <password_change_indicator>no_change</password_change_indicator>
       <password_change_date>01-07-2020</password_change_date>
       <shipping_address_usage_indicator>current_transaction</shipping_address_usage_indicator>
       <shipping_address_date_first_used>11-07-2020</shipping_address_date_first_used>
       <transactions_activity_last_24_hours>2</transactions_activity_last_24_hours>
       <transactions_activity_previous_year>10</transactions_activity_previous_year>
       <provision_attempts_last_24_hours>1</provision_attempts_last_24_hours>
       <purchases_count_last_6_months>5</purchases_count_last_6_months>
       <suspicious_activity_indicator>no_suspicious_observed</suspicious_activity_indicator>
       <registration_indicator>30_to_60_days</registration_indicator>
       <registration_date>16-07-2018</registration_date>
     </card_holder_account>
     <browser>
       <accept_header>Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser</accept_header>
       <java_enabled>true</java_enabled>
       <language>en-GB</language>
       <color_depth>32</color_depth>
       <screen_height>900</screen_height>
       <screen_width>1440</screen_width>
       <time_zone_offset>-120</time_zone_offset>
       <user_agent>Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.97 Safari/537.36</user_agent>
     </browser>
     <sdk>
       <interface>native</interface>
       <ui_types>
         <ui_type>multi_select</ui_type>
       </ui_types>
       <application_id>fc1650c0-5778-0138-8205-2cbc32a32d65</application_id>
       <encrypted_data>encrypted-data-here</encrypted_data>
       <ephemeral_public_key_pair>public-key-pair</ephemeral_public_key_pair>
       <max_timeout>10</max_timeout>
       <reference_number>sdk-reference-number-here</reference_number>
     </sdk>
   </threeds_v2_params>
 </payment_transaction>'
```

Asynchronous 3 D Sv2 Frictionless With 3ds Method Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Authorize3D');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('4066330000000004')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')
        ->setNotificationUrl('https://www.example.com/notification')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')


        // Threeds V2 Params
        ->setThreedsControl('{"device_type"="browser", "challenge_window_size"="full_screen", "challenge_indicator"="preference"}')
        ->setThreedsPurchase('{"category"="service"}')
        ->setThreedsRecurring('{"expiration_date"="16-01-2021", "frequency"=>30}')
        ->setThreedsMerchantRisk('{"shipping_indicator"="verified_address", "delivery_timeframe"="electronic", "reorder_items_indicator"="reordered", "pre_order_purchase_indicator"="merchandise_available",
        ->setThreedsCardHolderAccount('{"creation_date"="16-07-2019", "update_indicator"="more_than_60days", "last_change_date"="16-04-2020", "password_change_indicator"="no_change", "password_change_date"
        ->setThreedsBrowser('{"accept_header"="Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser", "java_enabled"="true", "language"="en-GB", "color_depth"="
        ->setThreedsSdk('{"interface"="native", "ui_types"={"ui_type"="multi_select"}, "application_id"="fc1650c0-5778-0138-8205-2cbc32a32d65", "encrypted_data"="encrypted-data-here", "ephemeral_public_ke

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.Authorize3DRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;
import java.net.MalformedURLException;
import java.net.URL;

public class GenesisExample {
  public static void main() throws MalformedURLException {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    Authorize3DRequest request = new Authorize3DRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("4066330000000004");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");
    request.setNotificationUrl(new URL("https://www.example.com/notification"));
    request.setReturnSuccessUrl(new URL("http://www.example.com/success"));
    request.setReturnFailureUrl(new URL("http://www.example.com/failure"));


    // Threeds V2 Params
    request.setThreedsControl("{"device_type"=>"browser", "challenge_window_size"=>"full_screen", "challenge_indicator"=>"preference"}");
    request.setThreedsPurchase("{"category"=>"service"}");
    request.setThreedsRecurring("{"expiration_date"=>"16-01-2021", "frequency"=>30}");
    request.setThreedsMerchantRisk("{"shipping_indicator"=>"verified_address", "delivery_timeframe"=>"electronic", "reorder_items_indicator"=>"reordered", "pre_order_purchase_indicator"=>"merchandise_available
    request.setThreedsCardHolderAccount("{"creation_date"=>"16-07-2019", "update_indicator"=>"more_than_60days", "last_change_date"=>"16-04-2020", "password_change_indicator"=>"no_change", "password_change_da
    request.setThreedsBrowser("{"accept_header"=>"Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser", "java_enabled"=>"true", "language"=>"en-GB", "color_depth"
    request.setThreedsSdk("{"interface"=>"native", "ui_types"=>{"ui_type"=>"multi_select"}, "application_id"=>"fc1650c0-5778-0138-8205-2cbc32a32d65", "encrypted_data"=>"encrypted-data-here", "ephemeral_public_

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.authorize3d(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "4066330000000004",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "billing_address": {
```

```
      "billing_address": {
        "first_name": "Travis",
        "last_name": "Pastrana",
        "address1": "Muster Str. 12",
        "zip_code": "10178",
        "city": "Los Angeles",
        "state": "CA",
        "country": "US"
      },
      "notification_url": "https://www.example.com/notification",
      "return_success_url": "http://www.example.com/success",
      "return_failure_url": "http://www.example.com/failure",
      "threeds_v2_params": {
        "control": {
          "device_type": "browser",
          "challenge_window_size": "full_screen",
          "challenge_indicator": "preference"
        },
        "purchase": {
          "category": "service"
        },
        "recurring": {
          "expiration_date": "16-01-2021",
          "frequency": 30
        },
        "merchant_risk": {
          "shipping_indicator": "verified_address",
          "delivery_timeframe": "electronic",
          "reorder_items_indicator": "reordered",
          "pre_order_purchase_indicator": "merchandise_available",
          "pre_order_date": "16-08-2020",
          "gift_card": "true",
          "gift_card_count": "2"
        },
        "card_holder_account": {
          "creation_date": "16-07-2019",
          "update_indicator": "more_than_60days",
          "last_change_date": "16-04-2020",
          "password_change_indicator": "no_change",
          "password_change_date": "01-07-2020",
          "shipping_address_usage_indicator": "current_transaction",
          "shipping_address_date_first_used": "11-07-2020",
          "transactions_activity_last_24_hours": "2",
          "transactions_activity_previous_year": "10",
          "provision_attempts_last_24_hours": "1",
          "purchases_count_last_6_months": "5",
          "suspicious_activity_indicator": "no_suspicious_observed",
          "registration_indicator": "30_to_60_days",
          "registration_date": "16-07-2018"
        },
        "browser": {
          "accept_header": "Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser",
          "java_enabled": "true",
          "language": "en-GB",
          "color_depth": "32",
          "screen_height": "900",
          "screen_width": "1440",
          "time_zone_offset": "-120",
          "user_agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.97 Safari/537.36"
        },
        "sdk": {
          "interface": "native",
          "ui_types": {
            "ui_type": "multi_select"
          },
          "application_id": "fc1650c0-5778-0138-8205-2cbc32a32d65",
          "encrypted_data": "encrypted-data-here",
          "ephemeral_public_key_pair": "public-key-pair",
          "max_timeout": "10",
          "reference_number": "sdk-reference-number-here"
        }
      }
    }
  ).send()
    .then(success)
    .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>authorize3d</transaction_type>
   <transaction_id>43671</transaction_id>
   <usage>40208 concert tickets</usage>
   <remote_ip>245.253.2.12</remote_ip>
   <amount>100</amount>
   <currency>USD</currency>
   <card_holder>Travis Pastrana</card_holder>
   <card_number>4066330000000004</card_number>
   <expiration_month>12</expiration_month>
   <expiration_year>2021</expiration_year>
   <cvv>834</cvv>
   <customer_email>travis@example.com</customer_email>
   <customer_phone>+1987987987987</customer_phone>
   <billing_address>
     <first_name>Travis</first_name>
     <last_name>Pastrana</last_name>
     <address1>Muster Str. 12</address1>
     <zip_code>10178</zip_code>
     <city>Los Angeles</city>
     <state>CA</state>
     <country>US</country>
   </billing_address>
   <notification_url>https://www.example.com/notification</notification_url>
   <return_success_url>http://www.example.com/success</return_success_url>
   <return_failure_url>http://www.example.com/failure</return_failure_url>
   <threeds_v2_params>
     <control>
       <device_type>browser</device_type>
       <challenge_window_size>full_screen</challenge_window_size>
       <challenge_indicator>preference</challenge_indicator>
     </control>
     <purchase>
       <category>service</category>
     </purchase>
     <merchant_risk>
       <shipping_indicator>verified_address</shipping_indicator>
       <delivery_timeframe>electronic</delivery_timeframe>
       <reorder_items_indicator>reordered</reorder_items_indicator>
       <pre_order_purchase_indicator>merchandise_available</pre_order_purchase_indicator>
       <pre_order_date>16-08-2020</pre_order_date>
       <gift_card>true</gift_card>
       <gift_card_count>2</gift_card_count>
     </merchant_risk>
     <card_holder_account>
       <creation_date>16-07-2019</creation_date>
       <update_indicator>more_than_60days</update_indicator>
       <last_change_date>16-04-2020</last_change_date>
       <password_change_indicator>no_change</password_change_indicator>
       <password_change_date>01-07-2020</password_change_date>
       <shipping_address_usage_indicator>current_transaction</shipping_address_usage_indicator>
       <shipping_address_date_first_used>11-07-2020</shipping_address_date_first_used>
       <transactions_activity_last_24_hours>2</transactions_activity_last_24_hours>
       <transactions_activity_previous_year>10</transactions_activity_previous_year>
       <provision_attempts_last_24_hours>1</provision_attempts_last_24_hours>
       <purchases_count_last_6_months>5</purchases_count_last_6_months>
       <suspicious_activity_indicator>no_suspicious_observed</suspicious_activity_indicator>
       <registration_indicator>30_to_60_days</registration_indicator>
       <registration_date>16-07-2018</registration_date>
     </card_holder_account>
     <browser>
       <accept_header>Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser</accept_header>
       <java_enabled>true</java_enabled>
       <language>en-GB</language>
       <color_depth>32</color_depth>
       <screen_height>900</screen_height>
       <screen_width>1440</screen_width>
       <time_zone_offset>-120</time_zone_offset>
       <user_agent>Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.97 Safari/537.36</user_agent>
     </browser>
     <sdk>
       <interface>native</interface>
       <ui_types>
         <ui_type>multi_select</ui_type>
       </ui_types>
       <application_id>fc1650c0-5778-0138-8205-2cbc32a32d65</application_id>
       <encrypted_data>encrypted-data-here</encrypted_data>
       <ephemeral_public_key_pair>public-key-pair</ephemeral_public_key_pair>
       <max_timeout>10</max_timeout>
       <reference_number>sdk-reference-number-here</reference_number>
     </sdk>
   </threeds_v2_params>
 </payment_transaction>'
```

Asynchronous 3 D Sv2 Challenge Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Authorize3D');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('4918190000000002')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')
        ->setNotificationUrl('https://www.example.com/notification')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')


        // Threeds V2 Params
        ->setThreedsControl('{"device_type"=>"browser", "challenge_window_size"=>"full_screen", "challenge_indicator"=>"preference"}')
        ->setThreedsPurchase('{"category"=>"service"}')
        ->setThreedsRecurring('{"expiration_date"=>"16-01-2021", "frequency"=>30}')
        ->setThreedsMerchantRisk('{"shipping_indicator"=>"verified_address", "delivery_timeframe"=>"electronic", "reorder_items_indicator"=>"reordered", "pre_order_purchase_indicator"=>"merchandise_available",
        ->setThreedsCardHolderAccount('{"creation_date"=>"16-07-2019", "update_indicator"=>"more_than_60days", "last_change_date"=>"16-04-2020", "password_change_indicator"=>"no_change", "password_change_date"
        ->setThreedsBrowser('{"accept_header"=>"Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser", "java_enabled"=>"true", "language"=>"en-GB", "color_depth"=>"
        ->setThreedsSdk('{"interface"=>"native", "ui_types"=>{"ui_type"=>"multi_select"}, "application_id"=>"fc1650c0-5778-0138-8205-2cbc32a32d65", "encrypted_data"=>"encrypted-data-here", "ephemeral_public_ke

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.Authorize3DRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;
import java.net.MalformedURLException;
import java.net.URL;

public class GenesisExample {
  public static void main() throws MalformedURLException {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    Authorize3DRequest request = new Authorize3DRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("4918190000000002");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");
    request.setNotificationUrl(new URL("https://www.example.com/notification"));
    request.setReturnSuccessUrl(new URL("http://www.example.com/success"));
    request.setReturnFailureUrl(new URL("http://www.example.com/failure"));


    // Threeds V2 Params
    request.setThreedsControl("{"device_type"=>"browser", "challenge_window_size"=>"full_screen", "challenge_indicator"=>"preference"}");
    request.setThreedsPurchase("{"category"=>"service"}");
    request.setThreedsRecurring("{"expiration_date"=>"16-01-2021", "frequency"=>30}");
    request.setThreedsMerchantRisk("{"shipping_indicator"=>"verified_address", "delivery_timeframe"=>"electronic", "reorder_items_indicator"=>"reordered", "pre_order_purchase_indicator"=>"merchandise_available
    request.setThreedsCardHolderAccount("{"creation_date"=>"16-07-2019", "update_indicator"=>"more_than_60days", "last_change_date"=>"16-04-2020", "password_change_indicator"=>"no_change", "password_change_da
    request.setThreedsBrowser("{"accept_header"=>"Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser", "java_enabled"=>"true", "language"=>"en-GB", "color_depth"
    request.setThreedsSdk("{"interface"=>"native", "ui_types"=>{"ui_type"=>"multi_select"}, "application_id"=>"fc1650c0-5778-0138-8205-2cbc32a32d65", "encrypted_data"=>"encrypted-data-here", "ephemeral_public_

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.authorize3d(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "4918190000000002",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "billing_address": {
```

```
      "billing_address": {
        "first_name": "Travis",
        "last_name": "Pastrana",
        "address1": "Muster Str. 12",
        "zip_code": "10178",
        "city": "Los Angeles",
        "state": "CA",
        "country": "US"
      },
      "notification_url": "https://www.example.com/notification",
      "return_success_url": "http://www.example.com/success",
      "return_failure_url": "http://www.example.com/failure",
      "threeds_v2_params": {
        "control": {
          "device_type": "browser",
          "challenge_window_size": "full_screen",
          "challenge_indicator": "preference"
        },
        "purchase": {
          "category": "service"
        },
        "recurring": {
          "expiration_date": "16-01-2021",
          "frequency": 30
        },
        "merchant_risk": {
          "shipping_indicator": "verified_address",
          "delivery_timeframe": "electronic",
          "reorder_items_indicator": "reordered",
          "pre_order_purchase_indicator": "merchandise_available",
          "pre_order_date": "16-08-2020",
          "gift_card": "true",
          "gift_card_count": "2"
        },
        "card_holder_account": {
          "creation_date": "16-07-2019",
          "update_indicator": "more_than_60days",
          "last_change_date": "16-04-2020",
          "password_change_indicator": "no_change",
          "password_change_date": "01-07-2020",
          "shipping_address_usage_indicator": "current_transaction",
          "shipping_address_date_first_used": "11-07-2020",
          "transactions_activity_last_24_hours": "2",
          "transactions_activity_previous_year": "10",
          "provision_attempts_last_24_hours": "1",
          "purchases_count_last_6_months": "5",
          "suspicious_activity_indicator": "no_suspicious_observed",
          "registration_indicator": "30_to_60_days",
          "registration_date": "16-07-2018"
        },
        "browser": {
          "accept_header": "Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser",
          "java_enabled": "true",
          "language": "en-GB",
          "color_depth": "32",
          "screen_height": "900",
          "screen_width": "1440",
          "time_zone_offset": "-120",
          "user_agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.97 Safari/537.36"
        },
        "sdk": {
          "interface": "native",
          "ui_types": {
            "ui_type": "multi_select"
          },
          "application_id": "fc1650c0-5778-0138-8205-2cbc32a32d65",
          "encrypted_data": "encrypted-data-here",
          "ephemeral_public_key_pair": "public-key-pair",
          "max_timeout": "10",
          "reference_number": "sdk-reference-number-here"
        }
      }
    }
  ).send()
    .then(success)
    .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>authorize3d</transaction_type>
   <transaction_id>43671</transaction_id>
   <usage>40208 concert tickets</usage>
   <remote_ip>245.253.2.12</remote_ip>
   <amount>100</amount>
   <currency>USD</currency>
   <card_holder>Travis Pastrana</card_holder>
   <card_number>4918190000000002</card_number>
   <expiration_month>12</expiration_month>
   <expiration_year>2021</expiration_year>
   <cvv>834</cvv>
   <customer_email>travis@example.com</customer_email>
   <customer_phone>+1987987987987</customer_phone>
   <billing_address>
     <first_name>Travis</first_name>
     <last_name>Pastrana</last_name>
     <address1>Muster Str. 12</address1>
     <zip_code>10178</zip_code>
     <city>Los Angeles</city>
     <state>CA</state>
     <country>US</country>
   </billing_address>
   <notification_url>https://www.example.com/notification</notification_url>
   <return_success_url>http://www.example.com/success</return_success_url>
   <return_failure_url>http://www.example.com/failure</return_failure_url>
   <threeds_v2_params>
     <control>
       <device_type>browser</device_type>
       <challenge_window_size>full_screen</challenge_window_size>
       <challenge_indicator>preference</challenge_indicator>
     </control>
     <purchase>
       <category>service</category>
     </purchase>
     <merchant_risk>
       <shipping_indicator>verified_address</shipping_indicator>
       <delivery_timeframe>electronic</delivery_timeframe>
       <reorder_items_indicator>reordered</reorder_items_indicator>
       <pre_order_purchase_indicator>merchandise_available</pre_order_purchase_indicator>
       <pre_order_date>16-08-2020</pre_order_date>
       <gift_card>true</gift_card>
       <gift_card_count>2</gift_card_count>
     </merchant_risk>
     <card_holder_account>
       <creation_date>16-07-2019</creation_date>
       <update_indicator>more_than_60days</update_indicator>
       <last_change_date>16-04-2020</last_change_date>
       <password_change_indicator>no_change</password_change_indicator>
       <password_change_date>01-07-2020</password_change_date>
       <shipping_address_usage_indicator>current_transaction</shipping_address_usage_indicator>
       <shipping_address_date_first_used>11-07-2020</shipping_address_date_first_used>
       <transactions_activity_last_24_hours>2</transactions_activity_last_24_hours>
       <transactions_activity_previous_year>10</transactions_activity_previous_year>
       <provision_attempts_last_24_hours>1</provision_attempts_last_24_hours>
       <purchases_count_last_6_months>5</purchases_count_last_6_months>
       <suspicious_activity_indicator>no_suspicious_observed</suspicious_activity_indicator>
       <registration_indicator>30_to_60_days</registration_indicator>
       <registration_date>16-07-2018</registration_date>
     </card_holder_account>
     <browser>
       <accept_header>Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser</accept_header>
       <java_enabled>true</java_enabled>
       <language>en-GB</language>
       <color_depth>32</color_depth>
       <screen_height>900</screen_height>
       <screen_width>1440</screen_width>
       <time_zone_offset>-120</time_zone_offset>
       <user_agent>Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.97 Safari/537.36</user_agent>
     </browser>
     <sdk>
       <interface>native</interface>
       <ui_types>
         <ui_type>multi_select</ui_type>
       </ui_types>
       <application_id>fc1650c0-5778-0138-8205-2cbc32a32d65</application_id>
       <encrypted_data>encrypted-data-here</encrypted_data>
       <ephemeral_public_key_pair>public-key-pair</ephemeral_public_key_pair>
       <max_timeout>10</max_timeout>
       <reference_number>sdk-reference-number-here</reference_number>
     </sdk>
   </threeds_v2_params>
 </payment_transaction>'
```

Asynchronous 3 D Sv2 Challenge With 3ds Method Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Authorize3D');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('4938730000000001')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')
        ->setNotificationUrl('https://www.example.com/notification')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')


        // Threeds V2 Params
        ->setThreedsControl('{"device_type"=>"browser", "challenge_window_size"=>"full_screen", "challenge_indicator"=>"preference"}')
        ->setThreedsPurchase('{"category"=>"service"}')
        ->setThreedsRecurring('{"expiration_date"=>"16-01-2021", "frequency"=>30}')
        ->setThreedsMerchantRisk('{"shipping_indicator"=>"verified_address", "delivery_timeframe"=>"electronic", "reorder_items_indicator"=>"reordered", "pre_order_purchase_indicator"=>"merchandise_available",
        ->setThreedsCardHolderAccount('{"creation_date"=>"16-07-2019", "update_indicator"=>"more_than_60days", "last_change_date"=>"16-04-2020", "password_change_indicator"=>"no_change", "password_change_date"
        ->setThreedsBrowser('{"accept_header"=>"Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser", "java_enabled"=>"true", "language"=>"en-GB", "color_depth"=>"
        ->setThreedsSdk('{"interface"=>"native", "ui_types"=>{"ui_type"=>"multi_select"}, "application_id"=>"fc1650c0-5778-0138-8205-2cbc32a32d65", "encrypted_data"=>"encrypted-data-here", "ephemeral_public_ke

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.Authorize3DRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;
import java.net.MalformedURLException;
import java.net.URL;

public class GenesisExample {
  public static void main() throws MalformedURLException {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    Authorize3DRequest request = new Authorize3DRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("4938730000000001");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");
    request.setNotificationUrl(new URL("https://www.example.com/notification"));
    request.setReturnSuccessUrl(new URL("http://www.example.com/success"));
    request.setReturnFailureUrl(new URL("http://www.example.com/failure"));


    // Threeds V2 Params
    request.setThreedsControl("{"device_type"=>"browser", "challenge_window_size"=>"full_screen", "challenge_indicator"=>"preference"}");
    request.setThreedsPurchase("{"category"=>"service"}");
    request.setThreedsRecurring("{"expiration_date"=>"16-01-2021", "frequency"=>30}");
    request.setThreedsMerchantRisk("{"shipping_indicator"=>"verified_address", "delivery_timeframe"=>"electronic", "reorder_items_indicator"=>"reordered", "pre_order_purchase_indicator"=>"merchandise_available
    request.setThreedsCardHolderAccount("{"creation_date"=>"16-07-2019", "update_indicator"=>"more_than_60days", "last_change_date"=>"16-04-2020", "password_change_indicator"=>"no_change", "password_change_da
    request.setThreedsBrowser("{"accept_header"=>"Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser", "java_enabled"=>"true", "language"=>"en-GB", "color_depth"
    request.setThreedsSdk("{"interface"=>"native", "ui_types"=>{"ui_type"=>"multi_select"}, "application_id"=>"fc1650c0-5778-0138-8205-2cbc32a32d65", "encrypted_data"=>"encrypted-data-here", "ephemeral_public_

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.authorize3d(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "4938730000000001",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "billing_address": {
```

```
      "billing_address": {
        "first_name": "Travis",
        "last_name": "Pastrana",
        "address1": "Muster Str. 12",
        "zip_code": "10178",
        "city": "Los Angeles",
        "state": "CA",
        "country": "US"
      },
      "notification_url": "https://www.example.com/notification",
      "return_success_url": "http://www.example.com/success",
      "return_failure_url": "http://www.example.com/failure",
      "threeds_v2_params": {
        "control": {
          "device_type": "browser",
          "challenge_window_size": "full_screen",
          "challenge_indicator": "preference"
        },
        "purchase": {
          "category": "service"
        },
        "recurring": {
          "expiration_date": "16-01-2021",
          "frequency": 30
        },
        "merchant_risk": {
          "shipping_indicator": "verified_address",
          "delivery_timeframe": "electronic",
          "reorder_items_indicator": "reordered",
          "pre_order_purchase_indicator": "merchandise_available",
          "pre_order_date": "16-08-2020",
          "gift_card": "true",
          "gift_card_count": "2"
        },
        "card_holder_account": {
          "creation_date": "16-07-2019",
          "update_indicator": "more_than_60days",
          "last_change_date": "16-04-2020",
          "password_change_indicator": "no_change",
          "password_change_date": "01-07-2020",
          "shipping_address_usage_indicator": "current_transaction",
          "shipping_address_date_first_used": "11-07-2020",
          "transactions_activity_last_24_hours": "2",
          "transactions_activity_previous_year": "10",
          "provision_attempts_last_24_hours": "1",
          "purchases_count_last_6_months": "5",
          "suspicious_activity_indicator": "no_suspicious_observed",
          "registration_indicator": "30_to_60_days",
          "registration_date": "16-07-2018"
        },
        "browser": {
          "accept_header": "Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser",
          "java_enabled": "true",
          "language": "en-GB",
          "color_depth": "32",
          "screen_height": "900",
          "screen_width": "1440",
          "time_zone_offset": "-120",
          "user_agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.97 Safari/537.36"
        },
        "sdk": {
          "interface": "native",
          "ui_types": {
            "ui_type": "multi_select"
          },
          "application_id": "fc1650c0-5778-0138-8205-2cbc32a32d65",
          "encrypted_data": "encrypted-data-here",
          "ephemeral_public_key_pair": "public-key-pair",
          "max_timeout": "10",
          "reference_number": "sdk-reference-number-here"
        }
      }
    }
  ).send()
    .then(success)
    .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>authorize3d</transaction_type>
   <transaction_id>43671</transaction_id>
   <usage>40208 concert tickets</usage>
   <remote_ip>245.253.2.12</remote_ip>
   <amount>100</amount>
   <currency>USD</currency>
   <card_holder>Travis Pastrana</card_holder>
   <card_number>4938730000000001</card_number>
   <expiration_month>12</expiration_month>
   <expiration_year>2021</expiration_year>
   <cvv>834</cvv>
   <customer_email>travis@example.com</customer_email>
   <customer_phone>+1987987987987</customer_phone>
   <billing_address>
     <first_name>Travis</first_name>
     <last_name>Pastrana</last_name>
     <address1>Muster Str. 12</address1>
     <zip_code>10178</zip_code>
     <city>Los Angeles</city>
     <state>CA</state>
     <country>US</country>
   </billing_address>
   <notification_url>https://www.example.com/notification</notification_url>
   <return_success_url>http://www.example.com/success</return_success_url>
   <return_failure_url>http://www.example.com/failure</return_failure_url>
   <threeds_v2_params>
     <control>
       <device_type>browser</device_type>
       <challenge_window_size>full_screen</challenge_window_size>
       <challenge_indicator>preference</challenge_indicator>
     </control>
     <purchase>
       <category>service</category>
     </purchase>
     <merchant_risk>
       <shipping_indicator>verified_address</shipping_indicator>
       <delivery_timeframe>electronic</delivery_timeframe>
       <reorder_items_indicator>reordered</reorder_items_indicator>
       <pre_order_purchase_indicator>merchandise_available</pre_order_purchase_indicator>
       <pre_order_date>16-08-2020</pre_order_date>
       <gift_card>true</gift_card>
       <gift_card_count>2</gift_card_count>
     </merchant_risk>
     <card_holder_account>
       <creation_date>16-07-2019</creation_date>
       <update_indicator>more_than_60days</update_indicator>
       <last_change_date>16-04-2020</last_change_date>
       <password_change_indicator>no_change</password_change_indicator>
       <password_change_date>01-07-2020</password_change_date>
       <shipping_address_usage_indicator>current_transaction</shipping_address_usage_indicator>
       <shipping_address_date_first_used>11-07-2020</shipping_address_date_first_used>
       <transactions_activity_last_24_hours>2</transactions_activity_last_24_hours>
       <transactions_activity_previous_year>10</transactions_activity_previous_year>
       <provision_attempts_last_24_hours>1</provision_attempts_last_24_hours>
       <purchases_count_last_6_months>5</purchases_count_last_6_months>
       <suspicious_activity_indicator>no_suspicious_observed</suspicious_activity_indicator>
       <registration_indicator>30_to_60_days</registration_indicator>
       <registration_date>16-07-2018</registration_date>
     </card_holder_account>
     <browser>
       <accept_header>Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser</accept_header>
       <java_enabled>true</java_enabled>
       <language>en-GB</language>
       <color_depth>32</color_depth>
       <screen_height>900</screen_height>
       <screen_width>1440</screen_width>
       <time_zone_offset>-120</time_zone_offset>
       <user_agent>Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.97 Safari/537.36</user_agent>
     </browser>
     <sdk>
       <interface>native</interface>
       <ui_types>
         <ui_type>multi_select</ui_type>
       </ui_types>
       <application_id>fc1650c0-5778-0138-8205-2cbc32a32d65</application_id>
       <encrypted_data>encrypted-data-here</encrypted_data>
       <ephemeral_public_key_pair>public-key-pair</ephemeral_public_key_pair>
       <max_timeout>10</max_timeout>
       <reference_number>sdk-reference-number-here</reference_number>
     </sdk>
   </threeds_v2_params>
 </payment_transaction>'
```

Asynchronous 3 D Sv2 Fallback 3dsv1 Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Authorize3D');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('4901170000000003')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')
        ->setNotificationUrl('https://www.example.com/notification')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')


        // Threeds V2 Params
        ->setThreedsControl('{"device_type"=>"browser", "challenge_window_size"=>"full_screen", "challenge_indicator"=>"preference"}')
        ->setThreedsPurchase('{"category"=>"service"}')
        ->setThreedsRecurring('{"expiration_date"=>"16-01-2021", "frequency"=>30}')
        ->setThreedsMerchantRisk('{"shipping_indicator"=>"verified_address", "delivery_timeframe"=>"electronic", "reorder_items_indicator"=>"reordered", "pre_order_purchase_indicator"=>"merchandise_available",
        ->setThreedsCardHolderAccount('{"creation_date"=>"16-07-2019", "update_indicator"=>"more_than_60days", "last_change_date"=>"16-04-2020", "password_change_indicator"=>"no_change", "password_change_date"
        ->setThreedsBrowser('{"accept_header"=>"Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser", "java_enabled"=>"true", "language"=>"en-GB", "color_depth"=>"
        ->setThreedsSdk('{"interface"=>"native", "ui_types"=>{"ui_type"=>"multi_select"}, "application_id"=>"fc1650c0-5778-0138-8205-2cbc32a32d65", "encrypted_data"=>"encrypted-data-here", "ephemeral_public_ke

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.Authorize3DRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;
import java.net.MalformedURLException;
import java.net.URL;

public class GenesisExample {
  public static void main() throws MalformedURLException {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    Authorize3DRequest request = new Authorize3DRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("4901170000000003");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");
    request.setNotificationUrl(new URL("https://www.example.com/notification"));
    request.setReturnSuccessUrl(new URL("http://www.example.com/success"));
    request.setReturnFailureUrl(new URL("http://www.example.com/failure"));


    // Threeds V2 Params
    request.setThreedsControl("{"device_type"=>"browser", "challenge_window_size"=>"full_screen", "challenge_indicator"=>"preference"}");
    request.setThreedsPurchase("{"category"=>"service"}");
    request.setThreedsRecurring("{"expiration_date"=>"16-01-2021", "frequency"=>30}");
    request.setThreedsMerchantRisk("{"shipping_indicator"=>"verified_address", "delivery_timeframe"=>"electronic", "reorder_items_indicator"=>"reordered", "pre_order_purchase_indicator"=>"merchandise_available
    request.setThreedsCardHolderAccount("{"creation_date"=>"16-07-2019", "update_indicator"=>"more_than_60days", "last_change_date"=>"16-04-2020", "password_change_indicator"=>"no_change", "password_change_da
    request.setThreedsBrowser("{"accept_header"=>"Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser", "java_enabled"=>"true", "language"=>"en-GB", "color_depth"
    request.setThreedsSdk("{"interface"=>"native", "ui_types"=>{"ui_type"=>"multi_select"}, "application_id"=>"fc1650c0-5778-0138-8205-2cbc32a32d65", "encrypted_data"=>"encrypted-data-here", "ephemeral_public_

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.authorize3d(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "4901170000000003",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "billing_address": {
```

```
      "billing_address": {
        "first_name": "Travis",
        "last_name": "Pastrana",
        "address1": "Muster Str. 12",
        "zip_code": "10178",
        "city": "Los Angeles",
        "state": "CA",
        "country": "US"
      },
      "notification_url": "https://www.example.com/notification",
      "return_success_url": "http://www.example.com/success",
      "return_failure_url": "http://www.example.com/failure",
      "threeds_v2_params": {
        "control": {
          "device_type": "browser",
          "challenge_window_size": "full_screen",
          "challenge_indicator": "preference"
        },
        "purchase": {
          "category": "service"
        },
        "recurring": {
          "expiration_date": "16-01-2021",
          "frequency": 30
        },
        "merchant_risk": {
          "shipping_indicator": "verified_address",
          "delivery_timeframe": "electronic",
          "reorder_items_indicator": "reordered",
          "pre_order_purchase_indicator": "merchandise_available",
          "pre_order_date": "16-08-2020",
          "gift_card": "true",
          "gift_card_count": "2"
        },
        "card_holder_account": {
          "creation_date": "16-07-2019",
          "update_indicator": "more_than_60days",
          "last_change_date": "16-04-2020",
          "password_change_indicator": "no_change",
          "password_change_date": "01-07-2020",
          "shipping_address_usage_indicator": "current_transaction",
          "shipping_address_date_first_used": "11-07-2020",
          "transactions_activity_last_24_hours": "2",
          "transactions_activity_previous_year": "10",
          "provision_attempts_last_24_hours": "1",
          "purchases_count_last_6_months": "5",
          "suspicious_activity_indicator": "no_suspicious_observed",
          "registration_indicator": "30_to_60_days",
          "registration_date": "16-07-2018"
        },
        "browser": {
          "accept_header": "Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser",
          "java_enabled": "true",
          "language": "en-GB",
          "color_depth": "32",
          "screen_height": "900",
          "screen_width": "1440",
          "time_zone_offset": "-120",
          "user_agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.97 Safari/537.36"
        },
        "sdk": {
          "interface": "native",
          "ui_types": {
            "ui_type": "multi_select"
          },
          "application_id": "fc1650c0-5778-0138-8205-2cbc32a32d65",
          "encrypted_data": "encrypted-data-here",
          "ephemeral_public_key_pair": "public-key-pair",
          "max_timeout": "10",
          "reference_number": "sdk-reference-number-here"
        }
      }
    }
  ).send()
    .then(success)
    .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>authorize3d</transaction_type>
   <transaction_id>43671</transaction_id>
   <usage>40208 concert tickets</usage>
   <remote_ip>245.253.2.12</remote_ip>
   <amount>100</amount>
   <currency>USD</currency>
   <card_holder>Travis Pastrana</card_holder>
   <card_number>4901170000000003</card_number>
   <expiration_month>12</expiration_month>
   <expiration_year>2021</expiration_year>
   <cvv>834</cvv>
   <customer_email>travis@example.com</customer_email>
   <customer_phone>+1987987987987</customer_phone>
   <billing_address>
     <first_name>Travis</first_name>
     <last_name>Pastrana</last_name>
     <address1>Muster Str. 12</address1>
     <zip_code>10178</zip_code>
     <city>Los Angeles</city>
     <state>CA</state>
     <country>US</country>
   </billing_address>
   <notification_url>https://www.example.com/notification</notification_url>
   <return_success_url>http://www.example.com/success</return_success_url>
   <return_failure_url>http://www.example.com/failure</return_failure_url>
   <threeds_v2_params>
     <control>
       <device_type>browser</device_type>
       <challenge_window_size>full_screen</challenge_window_size>
       <challenge_indicator>preference</challenge_indicator>
     </control>
     <purchase>
       <category>service</category>
     </purchase>
     <merchant_risk>
       <shipping_indicator>verified_address</shipping_indicator>
       <delivery_timeframe>electronic</delivery_timeframe>
       <reorder_items_indicator>reordered</reorder_items_indicator>
       <pre_order_purchase_indicator>merchandise_available</pre_order_purchase_indicator>
       <pre_order_date>16-08-2020</pre_order_date>
       <gift_card>true</gift_card>
       <gift_card_count>2</gift_card_count>
     </merchant_risk>
     <card_holder_account>
       <creation_date>16-07-2019</creation_date>
       <update_indicator>more_than_60days</update_indicator>
       <last_change_date>16-04-2020</last_change_date>
       <password_change_indicator>no_change</password_change_indicator>
       <password_change_date>01-07-2020</password_change_date>
       <shipping_address_usage_indicator>current_transaction</shipping_address_usage_indicator>
       <shipping_address_date_first_used>11-07-2020</shipping_address_date_first_used>
       <transactions_activity_last_24_hours>2</transactions_activity_last_24_hours>
       <transactions_activity_previous_year>10</transactions_activity_previous_year>
       <provision_attempts_last_24_hours>1</provision_attempts_last_24_hours>
       <purchases_count_last_6_months>5</purchases_count_last_6_months>
       <suspicious_activity_indicator>no_suspicious_observed</suspicious_activity_indicator>
       <registration_indicator>30_to_60_days</registration_indicator>
       <registration_date>16-07-2018</registration_date>
     </card_holder_account>
     <browser>
       <accept_header>Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser</accept_header>
       <java_enabled>true</java_enabled>
       <language>en-GB</language>
       <color_depth>32</color_depth>
       <screen_height>900</screen_height>
       <screen_width>1440</screen_width>
       <time_zone_offset>-120</time_zone_offset>
       <user_agent>Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.97 Safari/537.36</user_agent>
     </browser>
     <sdk>
       <interface>native</interface>
       <ui_types>
         <ui_type>multi_select</ui_type>
       </ui_types>
       <application_id>fc1650c0-5778-0138-8205-2cbc32a32d65</application_id>
       <encrypted_data>encrypted-data-here</encrypted_data>
       <ephemeral_public_key_pair>public-key-pair</ephemeral_public_key_pair>
       <max_timeout>10</max_timeout>
       <reference_number>sdk-reference-number-here</reference_number>
     </sdk>
   </threeds_v2_params>
 </payment_transaction>'
```

Asynchronous 3 D Sv2 Fallback 3dsv1 With 3ds Method Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Authorize3D');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('4901164281364345')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')
        ->setNotificationUrl('https://www.example.com/notification')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')


        // Threeds V2 Params
        ->setThreedsControl('{"device_type"=>"browser", "challenge_window_size"=>"full_screen", "challenge_indicator"=>"preference"}')
        ->setThreedsPurchase('{"category"=>"service"}')
        ->setThreedsRecurring('{"expiration_date"=>"16-01-2021", "frequency"=>30}')
        ->setThreedsMerchantRisk('{"shipping_indicator"=>"verified_address", "delivery_timeframe"=>"electronic", "reorder_items_indicator"=>"reordered", "pre_order_purchase_indicator"=>"merchandise_available",
        ->setThreedsCardHolderAccount('{"creation_date"=>"16-07-2019", "update_indicator"=>"more_than_60days", "last_change_date"=>"16-04-2020", "password_change_indicator"=>"no_change", "password_change_date"
        ->setThreedsBrowser('{"accept_header"=>"Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser", "java_enabled"=>"true", "language"=>"en-GB", "color_depth"=>"
        ->setThreedsSdk('{"interface"=>"native", "ui_types"=>{"ui_type"=>"multi_select"}, "application_id"=>"fc1650c0-5778-0138-8205-2cbc32a32d65", "encrypted_data"=>"encrypted-data-here", "ephemeral_public_ke

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.Authorize3DRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;
import java.net.MalformedURLException;
import java.net.URL;

public class GenesisExample {
  public static void main() throws MalformedURLException {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    Authorize3DRequest request = new Authorize3DRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("4901164281364345");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");
    request.setNotificationUrl(new URL("https://www.example.com/notification"));
    request.setReturnSuccessUrl(new URL("http://www.example.com/success"));
    request.setReturnFailureUrl(new URL("http://www.example.com/failure"));


    // Threeds V2 Params
    request.setThreedsControl("{"device_type"=>"browser", "challenge_window_size"=>"full_screen", "challenge_indicator"=>"preference"}");
    request.setThreedsPurchase("{"category"=>"service"}");
    request.setThreedsRecurring("{"expiration_date"=>"16-01-2021", "frequency"=>30}");
    request.setThreedsMerchantRisk("{"shipping_indicator"=>"verified_address", "delivery_timeframe"=>"electronic", "reorder_items_indicator"=>"reordered", "pre_order_purchase_indicator"=>"merchandise_available
    request.setThreedsCardHolderAccount("{"creation_date"=>"16-07-2019", "update_indicator"=>"more_than_60days", "last_change_date"=>"16-04-2020", "password_change_indicator"=>"no_change", "password_change_da
    request.setThreedsBrowser("{"accept_header"=>"Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser", "java_enabled"=>"true", "language"=>"en-GB", "color_depth"
    request.setThreedsSdk("{"interface"=>"native", "ui_types"=>{"ui_type"=>"multi_select"}, "application_id"=>"fc1650c0-5778-0138-8205-2cbc32a32d65", "encrypted_data"=>"encrypted-data-here", "ephemeral_public_

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.authorize3d(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "4901164281364345",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "billing_address": {
```

```
      "billing_address": {
        "first_name": "Travis",
        "last_name": "Pastrana",
        "address1": "Muster Str. 12",
        "zip_code": "10178",
        "city": "Los Angeles",
        "state": "CA",
        "country": "US"
      },
      "notification_url": "https://www.example.com/notification",
      "return_success_url": "http://www.example.com/success",
      "return_failure_url": "http://www.example.com/failure",
      "threeds_v2_params": {
        "control": {
          "device_type": "browser",
          "challenge_window_size": "full_screen",
          "challenge_indicator": "preference"
        },
        "purchase": {
          "category": "service"
        },
        "recurring": {
          "expiration_date": "16-01-2021",
          "frequency": 30
        },
        "merchant_risk": {
          "shipping_indicator": "verified_address",
          "delivery_timeframe": "electronic",
          "reorder_items_indicator": "reordered",
          "pre_order_purchase_indicator": "merchandise_available",
          "pre_order_date": "16-08-2020",
          "gift_card": "true",
          "gift_card_count": "2"
        },
        "card_holder_account": {
          "creation_date": "16-07-2019",
          "update_indicator": "more_than_60days",
          "last_change_date": "16-04-2020",
          "password_change_indicator": "no_change",
          "password_change_date": "01-07-2020",
          "shipping_address_usage_indicator": "current_transaction",
          "shipping_address_date_first_used": "11-07-2020",
          "transactions_activity_last_24_hours": "2",
          "transactions_activity_previous_year": "10",
          "provision_attempts_last_24_hours": "1",
          "purchases_count_last_6_months": "5",
          "suspicious_activity_indicator": "no_suspicious_observed",
          "registration_indicator": "30_to_60_days",
          "registration_date": "16-07-2018"
        },
        "browser": {
          "accept_header": "Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser",
          "java_enabled": "true",
          "language": "en-GB",
          "color_depth": "32",
          "screen_height": "900",
          "screen_width": "1440",
          "time_zone_offset": "-120",
          "user_agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.97 Safari/537.36"
        },
        "sdk": {
          "interface": "native",
          "ui_types": {
            "ui_type": "multi_select"
          },
          "application_id": "fc1650c0-5778-0138-8205-2cbc32a32d65",
          "encrypted_data": "encrypted-data-here",
          "ephemeral_public_key_pair": "public-key-pair",
          "max_timeout": "10",
          "reference_number": "sdk-reference-number-here"
        }
      }
    }
  ).send()
    .then(success)
    .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>authorize3d</transaction_type>
   <transaction_id>43671</transaction_id>
   <usage>40208 concert tickets</usage>
   <remote_ip>245.253.2.12</remote_ip>
   <amount>100</amount>
   <currency>USD</currency>
   <card_holder>Travis Pastrana</card_holder>
   <card_number>4901164281364345</card_number>
   <expiration_month>12</expiration_month>
   <expiration_year>2021</expiration_year>
   <cvv>834</cvv>
   <customer_email>travis@example.com</customer_email>
   <customer_phone>+1987987987987</customer_phone>
   <billing_address>
     <first_name>Travis</first_name>
     <last_name>Pastrana</last_name>
     <address1>Muster Str. 12</address1>
     <zip_code>10178</zip_code>
     <city>Los Angeles</city>
     <state>CA</state>
     <country>US</country>
   </billing_address>
   <notification_url>https://www.example.com/notification</notification_url>
   <return_success_url>http://www.example.com/success</return_success_url>
   <return_failure_url>http://www.example.com/failure</return_failure_url>
   <threeds_v2_params>
     <control>
       <device_type>browser</device_type>
       <challenge_window_size>full_screen</challenge_window_size>
       <challenge_indicator>preference</challenge_indicator>
     </control>
     <purchase>
       <category>service</category>
     </purchase>
     <merchant_risk>
       <shipping_indicator>verified_address</shipping_indicator>
       <delivery_timeframe>electronic</delivery_timeframe>
       <reorder_items_indicator>reordered</reorder_items_indicator>
       <pre_order_purchase_indicator>merchandise_available</pre_order_purchase_indicator>
       <pre_order_date>16-08-2020</pre_order_date>
       <gift_card>true</gift_card>
       <gift_card_count>2</gift_card_count>
     </merchant_risk>
     <card_holder_account>
       <creation_date>16-07-2019</creation_date>
       <update_indicator>more_than_60days</update_indicator>
       <last_change_date>16-04-2020</last_change_date>
       <password_change_indicator>no_change</password_change_indicator>
       <password_change_date>01-07-2020</password_change_date>
       <shipping_address_usage_indicator>current_transaction</shipping_address_usage_indicator>
       <shipping_address_date_first_used>11-07-2020</shipping_address_date_first_used>
       <transactions_activity_last_24_hours>2</transactions_activity_last_24_hours>
       <transactions_activity_previous_year>10</transactions_activity_previous_year>
       <provision_attempts_last_24_hours>1</provision_attempts_last_24_hours>
       <purchases_count_last_6_months>5</purchases_count_last_6_months>
       <suspicious_activity_indicator>no_suspicious_observed</suspicious_activity_indicator>
       <registration_indicator>30_to_60_days</registration_indicator>
       <registration_date>16-07-2018</registration_date>
     </card_holder_account>
     <browser>
       <accept_header>Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser</accept_header>
       <java_enabled>true</java_enabled>
       <language>en-GB</language>
       <color_depth>32</color_depth>
       <screen_height>900</screen_height>
       <screen_width>1440</screen_width>
       <time_zone_offset>-120</time_zone_offset>
       <user_agent>Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.97 Safari/537.36</user_agent>
     </browser>
     <sdk>
       <interface>native</interface>
       <ui_types>
         <ui_type>multi_select</ui_type>
       </ui_types>
       <application_id>fc1650c0-5778-0138-8205-2cbc32a32d65</application_id>
       <encrypted_data>encrypted-data-here</encrypted_data>
       <ephemeral_public_key_pair>public-key-pair</ephemeral_public_key_pair>
       <max_timeout>10</max_timeout>
       <reference_number>sdk-reference-number-here</reference_number>
     </sdk>
   </threeds_v2_params>
 </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **authorize3d** |

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | required* | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| gaming | optional | "true" | Signifies whether a gaming transaction is performed. Gaming transactions usually use MCC 7995. Contact tech-support@e-comprocessing.com for more details |
| preauthorization | optional | "true" | Signifies whether a preauthorization transaction is performed. Check the Preauthorizations section or contact tech support for more details. |
| moto | optional | "true" | Signifies whether a MOTO (mail order telephone order) transaction is performed. Contacttech-support@e-comprocessing.com for more details |
| crypto | optional | "true" | Signifies whether a purchase of crypto-currency transaction is performed. Must be populated when purchasing crypto-currency with a VISA card and MCC is 6051. Must be populated when purchasing crypto-currency with a MASTER or INTL MAESTRO card and MCC is one of 6051, 6211. Contact tech-support@e-comprocessing.com for more details |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| notification_url | required[1] | url | URL at merchant where gateway sends outcome of transaction. |
| return_success_url | required[1] | url | URL where customer is sent to after successful payment |
| return_failure_url | required[1] | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| card_holder | required | string(255) | Full name of customer as printed on credit card (first name and last name at least) |
| card_number | required | 13 to 16 digits | Complete cc number of customer |
| cvv | required | 3 to 4 digits | cvv of cc, requirement is based on terminal configuration |
| expiration_month | required | MM | Expiration month as printed on credit card |
| expiration_year | required | YYYY | Expiration year as printed on credit card |
| token | optional | string(36) | See Tokenization for more details. If present, the cardholder parameters can be omitted. Cannot be set together with `remember_card` |
| remember_card | optional | "true" | See Tokenize. Tokenizes cardholder parameters. Cannot be set together with `token` |
| consumer_id | optional | string(10) | See Consumers and Tokenization. Combine with `remember_card` to tokenize or with `token` to use token |
| **credential_on_file** | required* | | See Credential On File (COF) for more details |
| initial_customer_initiated | required* | string(18) | Initial transaction initiated by customer. Required for external tokenization, and optional for gateway-based tokenization |
| subsequent_customer_initiated | required* | string(18) | Subsequent transaction initiated by customer. Required for external tokenization, and optional for gateway-based tokenization |
| merchant_unscheduled | optional | string(20) | Transaction is initiated by the merchant |
| customer_email | required* | e-mail address | Must contain valid e-mail of customer |
| customer_phone | required* | string(32) | Must contain valid phone number of customer |
| birth_date | required* | dd-mm-yyyy | Required when MCC is a Financial Services one (e.g. MCC 6012) and either card brand is Visa or Mastercard/Maestro with UK-based merchant, UK-based bin (domestic), and DEBIT card type |
| document_id | required* | sring(20) | Document ID of the consumer. See Document ID Parameter for more details. |
| fx_rate_id | optional | integer | See Get rates for FX Service. Offers the option to use a specific FX rate to convert the transaction processing amount. Used FX rate should have the same source currency as the processing currency. Contact tech-support@e-comprocessing.com for more details |
| **business_attributes** | required* | | Check business attributes section. |
| event_start_date | required* | dd-mm-yyyy | The date when event starts in format dd-mm-yyyy |
| event_end_date | required* | dd-mm-yyyy | The date when event ends in format dd-mm-yyyy |
| event_organizer_id | required* | string | |
| event_id | required* | string | |
| date_of_order | required* | dd-mm-yyyy | The date when order was placed in format dd-mm-yyyy |

| Parameter | Required | Format | Description |
|---|---|---|---|
| delivery_date | required* | dd-mm-yyyy | Date of the expected delivery in format dd-mm-yyyy |
| name_of_the_supplier | required* | string | |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required* | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |
| **mpi_params** | required[2] | | |
| cavv | required[3] | string(255) | Verification Id of the authentication. Please note this can be the CAVV for Visa Card or UCAF to identify MasterCard. |
| eci | required[3] | string(255) | See Electronic Commerce Indicator as returned from the MPI for details |
| xid | required[3] | string(255) | Transaction ID generated by the 3DSv1 service that uniquely identifies the 3DSecure request. Note, it should not be submitted for transactions, which are though the 3DSv2 authentication protocol. |
| protocol_version | required[4] | string | The used 3DS protocol version. Default is 1 if not supplied. |
| directory_server_id | required[4] | string | The Directory Server ID used for 3DSecure transactions through the 3DSv2 authentication protocol. |
| **dynamic_descriptor_params** | optional | | |
| merchant_name | optional | string(25) | Allows to dynamically override the charge descriptor |
| merchant_city | optional | string(13) | Allows to dynamically override the merchant phone number |
| sub_merchant_id | optional | string(15) | Allows to dynamically override the sub-merchant ID. |
| **sca_params** | optional | | SCA params |
| exemption | optional | string | The exemption that the transaction should take advantage of. |
| visa_merchant_id | required[5] | string(8) | VMID assigned by Visa if participating in Trusted merchant program. |
| **threeds_v2_params** | required* | | 3DSv2 async parameters. They must be submitted in order to use the 3DSv2 authentication protocol in asynchronous workflow |
| **control** | required* | | General params for preferences in authentication flow and providing device interface information. |
| device_type | required* | string | Identifies the device channel of the consumer, **required** in the 3DSv2 authentication protocol. For more information, go to 3DSv2 control params |
| challenge_window_size | required* | string | Identifies the size of the challenge window for the consumer. For more information, go to3DSv2 control params |
| challenge_indicator | optional | string | The value has weight and might impact the decision whether a challenge will be required for the transaction or not. If not provided, it will be interpreted as **no_preference**. For more information, go to3DSv2 control params |
| **purchase** | required* | | Purchase related params providing with additional information regarding the order. |
| category | required* | string | Identifies the type of transaction being authenticated. This field is required in some markets.Accepted values are: **goods**, **service**, **check_acceptance**, **account_funding**, **quasi_cash**, **prepaid_activation**, **loan**. |

| Parameter | Required | Format | Description |
|---|---|---|---|
| **merchant_risk** | recommended | | Merchant risk assessment params. They are all optional, but recommended' |
| shipping_indicator | optional | string(16) | Indicator code that most accurately describes the shipping method for the cardholder specific transaction. If one or more items are included in the sale, use the Shipping Indicator code for the physical goods. If all digital goods, use the code that describes the most expensive item. Accepted values are: **same_as_billing**, **stored_address**, **verified_address**, **pick_up**, **digital_goods**, **travel**, **event_tickets**, **other**. |
| delivery_timeframe | optional | string(11) | Indicates the merchandise delivery timeframe. Accepted values are: **electronic**, **same_day**, **over_night**, **another_day**. |
| reorder_items_indicator | optional | string(10) | Indicates whether the cardholder is reordering previously purchased merchandise. Accepted values are: **first_time**, **reordered**. |
| pre_order_purchase_indicator | optional | string(21) | Indicates whether cardholder is placing an order for merchandise with a future-availability or release date. Accepted values are: **merchandise_available**, **future_availability**. |
| pre_order_date | optional | dd-mm-yyyy | For a pre-ordered purchase, the expected date that the merchandise will be available. |
| gift_card | optional | 'true' | Prepaid or gift card purchase. |
| gift_card_count | optional | integer | For prepaid or gift card purchase, total count of individual prepaid or gift cards/codes purchased. The value is limited to **99**. |
| **card_holder_account** | recommended | | Cardholder account additional information. They are all optional, but recommended, because they have a significant impact on approval rates |
| creation_date | optional | dd-mm-yyyy | Date that the cardholder opened the account with the 3DS Requester. |
| update_indicator | optional | string(19) | Length of time since the cardholder's account information with the 3DS Requestor was last changed. Includes Billing or Shipping address, new payment account, or new user(s) added. Accepted values are: **current_transaction**, **less_than_30days**, **30_to_60_days**, **more_than_60days**. |
| last_change_date | optional | dd-mm-yyyy | Date that the cardholder's account with the 3DS Requestor was last changed. Including Billing or Shipping address, new payment accout, or new user(s) added. |
| password_change_indicator | optional | string(18) | Length of time since the cardholder account with the 3DS Requestor had a password change or account reset. Accepted values are: **no_change**, **during_transaction**, **less_than_30days**, **30_to_60_days**, **more_than_60days**. |
| password_change_date | optional | dd-mm-yyyy | Date that cardholder's account with the 3DS Requestor had a password change or account reset. |
| shipping_address_usage_indicator | optional | string(19) | Indicates when the shipping address used for this transaction was first used with the 3DS Requestor. Accepted values are: **current_transaction**, **less_than_30days**, **30_to_60_days**, **more_than_60days**. |
| shipping_address_date_first_used | optional | dd-mm-yyyy | Date when the shipping address used for this transaction was first used with the 3DS Requestor. |
| transactions_activity_last_24_hours | optional | integer | Number of transactions (successful and abandoned) for this cardholder account with the 3DS Requestor across all payment accounts in the previous 24 hours. |
| transactions_activity_previous_year | optional | integer | Number of transactions (successful and abandoned) for this cardholder account with the 3DS Requestor across all payment accounts in the previous year. |
| provision_attempts_last_24_hours | optional | integer | Number of Add Card attempts in the last 24 hours. |
| purchases_count_last_6_months | optional | integer | Number of purchases with this cardholder account during the previous six months. |
| suspicious_activity_indicator | optional | string(22) | Indicates whether the 3DS Requestor has experienced suspicious activity (including previous fraud) on the cardholder account. Accepted values are: **no_suspicious_observed**, **suspicious_observed**. |
| registration_indicator | optional | string(19) | Indicates the length of time that the payment account was enrolled in the cardholder's account with the 3DS Requester. Accepted values are: **guest_checkout**, **current_transaction**, **less_than_30days**, **30_to_60_days**, **more_than_60days**. |
| registration_date | optional | dd-mm-yyyy | Date that the payment account was enrolled in the cardholder's account with the 3DS Requestor. |
| **browser** | required* | | For browser-based transactions. They are all *required* in case the **device_type** is set to **browser** |
| accept_header | required* | string(2048) | Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser. |
| java_enabled | required* | boolean | Boolean that represents the ability of the cardholder browser to execute Java. The value can be retrieved by accessing a property of the navigator with JavaScript, `navigator.javaEnabled`. |
| language | required* | string(8) | Value representing the browser language as defined in IETF BCP47. The value can be retrieved by accessing a property of the navigator with JavaScript, `navigator.language`. |
| color_depth | required* | integer | Value representing the bit depth of the colour palette for displaying images, in bits per pixel. Obtained from Cardholder browser using the `screen.colorDepth` property. The value can be one of **1**, **4**, **8**, **15**, **16**, **24**, **32**, **48** |
| screen_height | required* | integer | Total height of the Cardholder's screen in pixels. Value is returned from the screen.height property. |
| screen_width | required* | integer | Total width of the Cardholder's screen in pixels. Value is returned from the screen.width property. |
| time_zone_offset | required* | string(5) | Time difference between UTC time and the Cardholder browser local time, in minutes. The value can be retrieved using Javascript `getTimezoneOffset()` method. |
| user_agent | required* | string(2048) | Exact content of the HTTP user-agent header. |

| Parameter | Required | Format | Description |
|---|---|---|---|
| **sdk** | required* | | For application-based transactions. They are all *required* in case the **device_type** is set to **application** |
| interface | required* | string(6) | SDK Interface types that the device of the consumer supports for displaying specific challenge interfaces within the SDK. Accepted values are: **native**, **html**, **both**. |
| **ui_types** | required* | | Lists all UI types that the device of the consumer supports for displaying specific challenge interfaces within the SDK. |
| ui_type | required* | string(13) | UI type that the device of the consumer supports for displaying specific challenge interface. Accepted values are: **text**, **single_select**, **multi_select**, **out_of_bag**, **other_html**. |
| application_id | required* | string(36) | Universally unique ID created upon all installations and updates of the 3DS Requestor APp on a Customer Device. This will be newly generated and stored by the 3DS SDK for each installation or update. The field is limited to 36 characters and it shall have a canonical format as defined in IETF RFC 4122. |
| encrypted_data | required* | string(64000) | JWE Object as defined Section 6.2.2.1 containing data encrypted by the SDK for the DS to decrypt. The data will be present when sending to DS, but not present from DS to ACS. |
| ephemeral_public_key_pair | required* | string(256) | Public key component of the ephemeral key pair generated by the 3DS SDK and used to establish session keys between the 3DS SDK and ACS. In AReq, this data element is contained within the ACS Signed Content JWS Object. The field is limited to maximum 256 characters. |
| max_timeout | required* | integer | Indicates the maximum amount of time (in minutes) for all exchanges. The field shall have value greater or equals than 05. |
| reference_number | required* | string(32) | Identifies the vendor and version of the 3DS SDK that is integrated in a 3DS Requestor App, assigned by EMVCo when the 3DS SDK is approved. The field is limited to 32 characters. |

`required* = conditionally required`

`1 - required if mpi_params is not present, the transaction will be handled asynchronously. Not required if configured on Terminal or Merchant level. Contact [tech-support@#{email_domain_name}](mailto:tech-support@#{email_domain_name}) for more details.`

`2 - required if transaction should be handled synchronous.`

`3 - eci is always required if mpi_params is present. xid and cavv are not required for the 3D attempted only workflow, where the merchant does not have the xid/cavv, only the eci.`

`4 - protocol_version is only required if 3DSv2 is used. When missing 3DSv1 will be used. directory_server_id is mandatory when protocol_version is 2.`

`5 - visa_merchant_id is required when exemption value is trusted_merchant.`

Successful Synchronous Response

```
stdClass Object
(
    [transaction_type] => authorize3d
    [status] => approved
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [consumer_id] => 123456
    [token] => ee946db8-d7db-4bb7-b608-b65b153e127d
    [avs_response_code] => 5I
    [avs_response_text] => Response provided by issuer processor; Address information not verified
    [authorization_code] => 345678
    [retrieval_reference_number] => 016813015184
    [response_code] => 00
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:15.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
    [partial_approval] => true
    [scheme_transaction_identifier] => 019091214161031
)
```

```xml
<payment_response content=[
    <transaction_type content=[authorize3d]>
    <status content=[approved]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <consumer_id content=[123456]>
    <token content=[ee946db8-d7db-4bb7-b608-b65b153e127d]>
    <avs_response_code content=[5I]>
    <avs_response_text content=[Response provided by issuer processor; Address information not verified]>
    <authorization_code content=[345678]>
    <retrieval_reference_number content=[016813015184]>
    <response_code content=[00]>
    <timestamp content=[2020-07-15T12:08:15Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[true]>
    <partial_approval content=[true]>
    <scheme_transaction_identifier content=[019091214161031]>
]>
```

```json
{
    transaction_type: "authorize3d",
    status: "approved",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    consumer_id: "123456",
    token: "ee946db8-d7db-4bb7-b608-b65b153e127d",
    avs_response_code: "5I",
    avs_response_text: "Response provided by issuer processor; Address information not verified",
    authorization_code: "345678",
    retrieval_reference_number: "016813015184",
    response_code: "00",
    timestamp: "2020-07-15T12:08:15Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "true",
    partial_approval: "true",
    scheme_transaction_identifier: "019091214161031",
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>authorize3d</transaction_type>
  <status>approved</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <consumer_id>123456</consumer_id>
  <token>ee946db8-d7db-4bb7-b608-b65b153e127d</token>
  <avs_response_code>5I</avs_response_code>
  <avs_response_text>Response provided by issuer processor; Address information not verified</avs_response_text>
  <authorization_code>345678</authorization_code>
  <retrieval_reference_number>016813015184</retrieval_reference_number>
  <response_code>00</response_code>
  <timestamp>2020-07-15T12:08:15Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
  <partial_approval>true</partial_approval>
  <scheme_transaction_identifier>019091214161031</scheme_transaction_identifier>
</payment_transaction>
```


Successful Asynchronous Response

```
stdClass Object
(
    [transaction_type] => authorize3d
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [consumer_id] => 123456
    [token] => ee946db8-d7db-4bb7-b608-b65b153e127d
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/threeds/authentication/fd3bbd0509d8ac3ef0708b76220b2509
    [redirect_url_type] => 3ds_v1_payer_authentication
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:15.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
    [scheme_transaction_identifier] => 019091214161031
)
```

```
<payment_response content=[
    <transaction_type content=[authorize3d]>
    <status content=[pending_async]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <consumer_id content=[123456]>
    <token content=[ee946db8-d7db-4bb7-b608-b65b153e127d]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <redirect_url content=[https://staging.gate.e-comprocessing.net/threeds/authentication/fd3bbd0509d8ac3ef0708b76220b2509]>
    <redirect_url_type content=[3ds_v1_payer_authentication]>
    <timestamp content=[2020-07-15T12:08:15Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[true]>
    <scheme_transaction_identifier content=[019091214161031]>
]>
```

```
{
    transaction_type: "authorize3d",
    status: "pending_async",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    consumer_id: "123456",
    token: "ee946db8-d7db-4bb7-b608-b65b153e127d",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    redirect_url: "https://staging.gate.e-comprocessing.net/threeds/authentication/fd3bbd0509d8ac3ef0708b76220b2509",
    redirect_url_type: "3ds_v1_payer_authentication",
    timestamp: "2020-07-15T12:08:15Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "true",
    scheme_transaction_identifier: "019091214161031",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>authorize3d</transaction_type>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <consumer_id>123456</consumer_id>
  <token>ee946db8-d7db-4bb7-b608-b65b153e127d</token>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <redirect_url>https://staging.gate.e-comprocessing.net/threeds/authentication/fd3bbd0509d8ac3ef0708b76220b2509</redirect_url>
  <redirect_url_type>3ds_v1_payer_authentication</redirect_url_type>
  <timestamp>2020-07-15T12:08:15Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
  <scheme_transaction_identifier>019091214161031</scheme_transaction_identifier>
</payment_transaction>
```


Frictionless / Challenge With Synchronous 3ds Method Response

```
stdClass Object
(
    [transaction_type] => authorize3d
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [consumer_id] => 123456
    [token] => ee946db8-d7db-4bb7-b608-b65b153e127d
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/threeds/threeds_method/1a096da019a4679a828baa175a306ae6
    [redirect_url_type] => 3ds_v2_threeds_method
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:15.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
    [scheme_transaction_identifier] => 019091214161031
    [authorization_code] => 345678
    [retrieval_reference_number] => 016813015184
)
```

```
<payment_response content=[
    <transaction_type content=[authorize3d]>
    <status content=[pending_async]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <consumer_id content=[123456]>
    <token content=[ee946db8-d7db-4bb7-b608-b65b153e127d]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <redirect_url content=[https://staging.gate.e-comprocessing.net/threeds/threeds_method/1a096da019a4679a828baa175a306ae6]>
    <redirect_url_type content=[3ds_v2_threeds_method]>
    <timestamp content=[2020-07-15T12:08:15Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[true]>
    <scheme_transaction_identifier content=[019091214161031]>
    <authorization_code content=[345678]>
    <retrieval_reference_number content=[016813015184]>
]>
```

```
{
    transaction_type: "authorize3d",
    status: "pending_async",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    consumer_id: "123456",
    token: "ee946db8-d7db-4bb7-b608-b65b153e127d",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    redirect_url: "https://staging.gate.e-comprocessing.net/threeds/threeds_method/1a096da019a4679a828baa175a306ae6",
    redirect_url_type: "3ds_v2_threeds_method",
    timestamp: "2020-07-15T12:08:15Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "true",
    scheme_transaction_identifier: "019091214161031",
    authorization_code: "345678",
    retrieval_reference_number: "016813015184",
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>authorize3d</transaction_type>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <consumer_id>123456</consumer_id>
  <token>ee946db8-d7db-4bb7-b608-b65b153e127d</token>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <redirect_url>https://staging.gate.e-comprocessing.net/threeds/threeds_method/1a096da019a4679a828baa175a306ae6</redirect_url>
  <redirect_url_type>3ds_v2_threeds_method</redirect_url_type>
  <timestamp>2020-07-15T12:08:15Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
  <scheme_transaction_identifier>019091214161031</scheme_transaction_identifier>
  <authorization_code>345678</authorization_code>
  <retrieval_reference_number>016813015184</retrieval_reference_number>
</payment_transaction>
```


Frictionless / Challenge With Asynchronous 3ds Method Response


```
stdClass Object
(
    [transaction_type] => authorize3d
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [consumer_id] => 123456
    [token] => ee946db8-d7db-4bb7-b608-b65b153e127d
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/threeds/threeds_method/1a096da019a4679a828baa175a306ae6/continue
    [redirect_url_type] => 3ds_v2_threeds_method
    [threeds_method_url] => https://staging.gate.e-comprocessing.net/threeds/threeds_method/649e1ff35c61
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:15.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
    [scheme_transaction_identifier] => 019091214161031
    [authorization_code] => 345678
    [retrieval_reference_number] => 016813015184
)
```



```
<payment_response content=[
    <transaction_type content=[authorize3d]>
    <status content=[pending_async]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <consumer_id content=[123456]>
    <token content=[ee946db8-d7db-4bb7-b608-b65b153e127d]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <redirect_url content=[https://staging.gate.e-comprocessing.net/threeds/threeds_method/1a096da019a4679a828baa175a306ae6/continue]>
    <redirect_url_type content=[3ds_v2_threeds_method]>
    <threeds_method_url content=[https://staging.gate.e-comprocessing.net/threeds/threeds_method/649e1ff35c61]>
    <timestamp content=[2020-07-15T12:08:15Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[true]>
    <scheme_transaction_identifier content=[019091214161031]>
    <authorization_code content=[345678]>
    <retrieval_reference_number content=[016813015184]>
]>
```

```
{
    transaction_type: "authorize3d",
    status: "pending_async",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    consumer_id: "123456",
    token: "ee946db8-d7db-4bb7-b608-b65b153e127d",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    redirect_url: "https://staging.gate.e-comprocessing.net/threeds/threeds_method/1a096da019a4679a828baa175a306ae6/continue",
    redirect_url_type: "3ds_v2_threeds_method",
    threeds_method_url: "https://staging.gate.e-comprocessing.net/threeds/threeds_method/649e1ff35c61",
    timestamp: "2020-07-15T12:08:15Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "true",
    scheme_transaction_identifier: "019091214161031",
    authorization_code: "345678",
    retrieval_reference_number: "016813015184",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>authorize3d</transaction_type>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <consumer_id>123456</consumer_id>
  <token>ee946db8-d7db-4bb7-b608-b65b153e127d</token>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <redirect_url>https://staging.gate.e-comprocessing.net/threeds/threeds_method/1a096da019a4679a828baa175a306ae6/continue</redirect_url>
  <redirect_url_type>3ds_v2_threeds_method</redirect_url_type>
  <threeds_method_url>https://staging.gate.e-comprocessing.net/threeds/threeds_method/649e1ff35c61</threeds_method_url>
  <timestamp>2020-07-15T12:08:15Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
  <scheme_transaction_identifier>019091214161031</scheme_transaction_identifier>
  <authorization_code>345678</authorization_code>
  <retrieval_reference_number>016813015184</retrieval_reference_number>
</payment_transaction>
```


Challenge Without 3 Ds Method Response


```
stdClass Object
(
    [transaction_type] => authorize3d
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [consumer_id] => 123456
    [token] => ee946db8-d7db-4bb7-b608-b65b153e127d
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/threeds/authentication/fd3bbd0509d8ac3ef0708b76220b2509
    [redirect_url_type] => 3ds_v2_challenge
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:15.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
    [scheme_transaction_identifier] => 019091214161031
)
```

```
<payment_response content=[
    <transaction_type content=[authorize3d]>
    <status content=[pending_async]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <consumer_id content=[123456]>
    <token content=[ee946db8-d7db-4bb7-b608-b65b153e127d]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <redirect_url content=[https://staging.gate.e-comprocessing.net/threeds/authentication/fd3bbd0509d8ac3ef0708b76220b2509]>
    <redirect_url_type content=[3ds_v2_challenge]>
    <timestamp content=[2020-07-15T12:08:15Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[true]>
    <scheme_transaction_identifier content=[019091214161031]>
]>
```

```
{
    transaction_type: "authorize3d",
    status: "pending_async",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    consumer_id: "123456",
    token: "ee946db8-d7db-4bb7-b608-b65b153e127d",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    redirect_url: "https://staging.gate.e-comprocessing.net/threeds/authentication/fd3bbd0509d8ac3ef0708b76220b2509",
    redirect_url_type: "3ds_v2_challenge",
    timestamp: "2020-07-15T12:08:15Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "true",
    scheme_transaction_identifier: "019091214161031",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>authorize3d</transaction_type>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <consumer_id>123456</consumer_id>
  <token>ee946db8-d7db-4bb7-b608-b65b153e127d</token>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <redirect_url>https://staging.gate.e-comprocessing.net/threeds/authentication/fd3bbd0509d8ac3ef0708b76220b2509</redirect_url>
  <redirect_url_type>3ds_v2_challenge</redirect_url_type>
  <timestamp>2020-07-15T12:08:15Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
  <scheme_transaction_identifier>019091214161031</scheme_transaction_identifier>
</payment_transaction>
```


Fallback To 3ds1 Without 3ds Method Response


```
stdClass Object
(
    [transaction_type] => authorize3d
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [consumer_id] => 123456
    [token] => ee946db8-d7db-4bb7-b608-b65b153e127d
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/threeds/authentication/fd3bbd0509d8ac3ef0708b76220b2509
    [redirect_url_type] => 3ds_v1_payer_authentication
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:15.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
    [scheme_transaction_identifier] => 019091214161031
)
```

```xml
<payment_response content=[
    <transaction_type content=[authorize3d]>
    <status content=[pending_async]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <consumer_id content=[123456]>
    <token content=[ee946db8-d7db-4bb7-b608-b65b153e127d]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <redirect_url content=[https://staging.gate.e-comprocessing.net/threeds/authentication/fd3bbd0509d8ac3ef0708b76220b2509]>
    <redirect_url_type content=[3ds_v1_payer_authentication]>
    <timestamp content=[2020-07-15T12:08:15Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[true]>
    <scheme_transaction_identifier content=[019091214161031]>
]>
```

```json
{
    transaction_type: "authorize3d",
    status: "pending_async",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    consumer_id: "123456",
    token: "ee946db8-d7db-4bb7-b608-b65b153e127d",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    redirect_url: "https://staging.gate.e-comprocessing.net/threeds/authentication/fd3bbd0509d8ac3ef0708b76220b2509",
    redirect_url_type: "3ds_v1_payer_authentication",
    timestamp: "2020-07-15T12:08:15Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "true",
    scheme_transaction_identifier: "019091214161031",
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>authorize3d</transaction_type>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <consumer_id>123456</consumer_id>
  <token>ee946db8-d7db-4bb7-b608-b65b153e127d</token>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <redirect_url>https://staging.gate.e-comprocessing.net/threeds/authentication/fd3bbd0509d8ac3ef0708b76220b2509</redirect_url>
  <redirect_url_type>3ds_v1_payer_authentication</redirect_url_type>
  <timestamp>2020-07-15T12:08:15Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
  <scheme_transaction_identifier>019091214161031</scheme_transaction_identifier>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| consumer_id | string(10) | Consumer unique reference. See Consumers |
| token | string(36) | Plain-text token value. See Tokenization |
| gaming | string(255) | Signifies whether a gaming transaction is performed. Gaming transactions usually use MCC 7995, contact tech support for more details. |
| moto | string(255) | Signifies whether a MOTO (mail order telephone order) transaction is performed. Contact tech support for more details. |
| avs_response_code | string(255) | Generated by the card network on trying to match the billing address when performing the address verification. Optional, returned if config is enabled and acquirer supports it. Check AVS Status Codes for details. |
| avs_response_text | string(255) | Gives the human response text for the AVS response code above. Optional, returned if config is enabled and acquirer supports it. Check AVS Status Codes for details. |
| authorization_code | string(6) | Generated by the card network when an authorisation has occurred, used to identify that auth. Consists of 6 alphanumeric chars |
| retrieval_reference_number | string(255) | A reference number used for tracking all messages related to a given cardholder transaction returned by some acquirers. |

| Parameter | Type | Description |
|---|---|---|
| response_code | string(2) | Defines the issuer result of a transaction, the status of a message or some action taken or required. See issuer response codes for details |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| redirect_url | url | URL where the consumer has to be redirected to complete payment process unless a 3DSecure Method is required. For more information, to go 3DSv2 authentication flows |
| redirect_url_type | string(64) | The type of the redirect URL. It might be present only for asynchronous 3D transactions For more information, to go 3DSv2 authentication flows |
| threeds_method_url | url | 3DSecure Method URL. It will be present only then asynchronous 3DSecure Method is required for 3D transaction. For more information, to go 3DSv2 authentication flows |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08- 30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| partial_approval | string(4) | Optional, set to 'true' if partial approval happened. Partially approved amount is then in the amount field. Check Partial Approvals for details |
| sent_to_acquirer | string(255) | "true" or "false" |
| dynamic_descriptor_params | section | Optional, returned only if dynamic desc params are submitted on the API. Note here that the formatted dyn desc params are returned - as they would be submitted to the schemes for settlement. |
| scheme_transaction_identifier | string(32) | Id defined by card schemes |

Error Response

```
stdClass Object
(
    [transaction_type] => authorize3d
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [response_code] => 57
    [code] => 340
    [technical_message] => expiration_year is invalid
    [message] => expiration_year is invalid
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:15.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => false
)
```

```
<payment_response content=[
    <transaction_type content=[authorize3d]>
    <status content=[error]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <response_code content=[57]>
    <code content=[340]>
    <technical_message content=[expiration_year is invalid]>
    <message content=[expiration_year is invalid]>
    <timestamp content=[2020-07-15T12:08:15Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[false]>
]>
```

```
{
    transaction_type: "authorize3d",
    status: "error",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    response_code: "57",
    code: "340",
    technical_message: "expiration_year is invalid",
    message: "expiration_year is invalid",
    timestamp: "2020-07-15T12:08:15Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "false",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>authorize3d</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <response_code>57</response_code>
  <code>340</code>
  <technical_message>expiration_year is invalid</technical_message>
  <message>expiration_year is invalid</message>
  <timestamp>2020-07-15T12:08:15Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>false</sent_to_acquirer>
</payment_transaction>
```


**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| gaming | string(255) | Signifies whether a gaming transaction is performed. Gaming transactions usually use MCC 7995, contact tech support for more details. |
| moto | string(255) | Signifies whether a MOTO (mail order telephone order) transaction is performed. Contact tech support for more details. |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08- 30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| dynamic_descriptor_params | section | Optional, returned only if dynamic desc params are submitted on the API. Note here that the formatted dyn desc params are returned - as they would be submitted to the schemes for settlement. |

#### SALE 3D

Sale3D transactions basically have the same request as standard sale transactions.

Sale3D transactions can be handled **synchronous** or **asynchronous** depending on the parameers passed. If mpi_params is passed the workflow will be **synchronous**. If **notification_url**, **return_success_url** and **return_failure_url** are passed the workflow will be **asynchronous**.

This transaction type supports Tokenization.

This transaction type supports Level 3 travel data.

This transaction type could require business attributes.

Visa Synchronous 3 D Sv1 Fully Authenticated Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Sale3D');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('4711100000000000')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')


        // MPI Synchronous 3DSv1
        ->setMpiEci('05')
        ->setMpiCavv('MDAwMDAwMDAwMDAxMTA2Njk5NFg=')
        ->setMpiXid('0pv62FIrT5qQODB7DCewKgEBAQI=')
        ->setMpiProtocolVersion('1')


        // Sca Params
        ->setScaExemption('trusted_merchant')
        ->setScaVisaMerchantId('00000000');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.Sale3DRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    Sale3DRequest request = new Sale3DRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("4711100000000000");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");


    // Mpi Synchronous V1
    request.setMpiParams("{"eci"=>"05", "cavv"=>"MDAwMDAwMDAwMDAxMTA2Njk5NFg=", "xid"=>"0pv62FIrT5qQODB7DCewKgEBAQI=", "protocol_version"=>"1"}");


    // Sca Params
    request.setScaExemption("trusted_merchant");
    request.setScaVisaMerchantId("00000000");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.sale3d(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "4711100000000000",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "business_attributes": {
      "event_start_date": "16-08-2020",
      "event_end_date": "25-08-2020",
      "event_organizer_id": "20192375",
      "event_id": "1912"
    },
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    },
    "mpi_params": {
      "eci": "05",
      "cavv": "MDAwMDAwMDAwMDAxMTA2Njk5NFg=",
      "xid": "0pv62FIrT5qQODB7DCewKgEBAQI=",
      "protocol_version": "1"
    },
    "sca_params": {
      "exemption": "trusted_merchant",
      "visa_merchant_id": "00000000"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>sale3d</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>100</amount>
    <currency>USD</currency>
    <card_holder>Travis Pastrana</card_holder>
    <card_number>4711100000000000</card_number>
    <expiration_month>12</expiration_month>
    <expiration_year>2021</expiration_year>
    <cvv>834</cvv>
    <customer_email>travis@example.com</customer_email>
    <customer_phone>+1987987987987</customer_phone>
    <business_attributes>
      <event_start_date>16-08-2020</event_start_date>
      <event_end_date>25-08-2020</event_end_date>
      <event_organizer_id>20192375</event_organizer_id>
      <event_id>1912</event_id>
    </business_attributes>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>10178</zip_code>
      <city>Los Angeles</city>
      <state>CA</state>
      <country>US</country>
    </billing_address>
    <mpi_params>
      <eci>05</eci>
      <cavv>MDAwMDAwMDAwMDAxMTA2Njk5NFg=</cavv>
      <xid>0pv62FIrT5qQODB7DCewKgEBAQI=</xid>
      <protocol_version>1</protocol_version>
    </mpi_params>
    <sca_params>
      <exemption>trusted_merchant</exemption>
      <visa_merchant_id>00000000</visa_merchant_id>
    </sca_params>
  </payment_transaction>'
```

Master Synchronous 3 D Sv1 Fully Authenticated Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Sale3D');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('5555555555559997')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')


        // MPI Synchronous 3DSv1
        ->setMpiEci('02')
        ->setMpiCavv('jlk/+AMJkDARYwAAA2SqCUQAAAA=')
        ->setMpiXid('UVg1QThJU1FVV1hWWVQxQkxZNU8=')
        ->setMpiProtocolVersion('1');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.Sale3DRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    Sale3DRequest request = new Sale3DRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("5555555555559997");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");


    // Mpi Synchronous V1
    request.setMpiParams("{"eci"=>"02", "cavv"=>"jlk/+AMJkDARYwAAA2SqCUQAAAA=", "xid"=>"UVg1QThJU1FVV1hWVQxQkxZNU8=", "protocol_version"=>"1"}");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.sale3d(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "5555555555559997",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "business_attributes": {
      "event_start_date": "16-08-2020",
      "event_end_date": "25-08-2020",
      "event_organizer_id": "20192375",
      "event_id": "1912"
    },
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    },
    "mpi_params": {
      "eci": "02",
      "cavv": "jlk/+AMJkDARYwAAA2SqCUQAAAA=",
      "xid": "UVg1QThJU1FVV1hWWVQxQkxZNU8=",
      "protocol_version": "1"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>sale3d</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>100</amount>
    <currency>USD</currency>
    <card_holder>Travis Pastrana</card_holder>
    <card_number>5555555555559997</card_number>
    <expiration_month>12</expiration_month>
    <expiration_year>2021</expiration_year>
    <cvv>834</cvv>
    <customer_email>travis@example.com</customer_email>
    <customer_phone>+1987987987987</customer_phone>
    <business_attributes>
      <event_start_date>16-08-2020</event_start_date>
      <event_end_date>25-08-2020</event_end_date>
      <event_organizer_id>20192375</event_organizer_id>
      <event_id>1912</event_id>
    </business_attributes>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>10178</zip_code>
      <city>Los Angeles</city>
      <state>CA</state>
      <country>US</country>
    </billing_address>
    <mpi_params>
      <eci>02</eci>
      <cavv>jlk/+AMJkDARYwAAA2SqCUQAAAA=</cavv>
      <xid>UVg1QThJU1FVV1hWWVQxQkxZNU8=</xid>
      <protocol_version>1</protocol_version>
    </mpi_params>
  </payment_transaction>'
```

Visa Synchronous 3 D Sv1 Attempted Authentication Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Sale3D');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('4711100000000000')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')


        // MPI Synchronous 3DSv1
        ->setMpiEci('06')
        ->setMpiCavv('MDAwMDAwMDAwMDAxMTA2Njk5NFg=')
        ->setMpiXid('0pv62FIrT5qQODB7DCewKgEBAQI=')
        ->setMpiProtocolVersion('1')


        // Sca Params
        ->setScaExemption('trusted_merchant')
        ->setScaVisaMerchantId('00000000');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.Sale3DRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    Sale3DRequest request = new Sale3DRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("4711100000000000");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");


    // Mpi Synchronous V1
    request.setMpiParams("{"eci"=>"06", "cavv"=>"MDAwMDAwMDAwMDAxMTA2Njk5NFg=", "xid"=>"0pv62FIrT5qQODB7DCewKgEBAQI=", "protocol_version"=>"1"}");


    // Sca Params
    request.setScaExemption("trusted_merchant");
    request.setScaVisaMerchantId("00000000");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.sale3d(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "4711100000000000",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "business_attributes": {
      "event_start_date": "16-08-2020",
      "event_end_date": "25-08-2020",
      "event_organizer_id": "20192375",
      "event_id": "1912"
    },
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    },
    "mpi_params": {
      "eci": "06",
      "cavv": "MDAwMDAwMDAwMDAxMTA2Njk5NFg=",
      "xid": "0pv62FIrT5qQODB7DCewKgEBAQI=",
      "protocol_version": "1"
    },
    "sca_params": {
      "exemption": "trusted_merchant",
      "visa_merchant_id": "00000000"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>sale3d</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>100</amount>
    <currency>USD</currency>
    <card_holder>Travis Pastrana</card_holder>
    <card_number>4711100000000000</card_number>
    <expiration_month>12</expiration_month>
    <expiration_year>2021</expiration_year>
    <cvv>834</cvv>
    <customer_email>travis@example.com</customer_email>
    <customer_phone>+1987987987987</customer_phone>
    <business_attributes>
      <event_start_date>16-08-2020</event_start_date>
      <event_end_date>25-08-2020</event_end_date>
      <event_organizer_id>20192375</event_organizer_id>
      <event_id>1912</event_id>
    </business_attributes>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>10178</zip_code>
      <city>Los Angeles</city>
      <state>CA</state>
      <country>US</country>
    </billing_address>
    <mpi_params>
      <eci>06</eci>
      <cavv>MDAwMDAwMDAwMDAxMTA2Njk5NFg=</cavv>
      <xid>0pv62FIrT5qQODB7DCewKgEBAQI=</xid>
      <protocol_version>1</protocol_version>
    </mpi_params>
    <sca_params>
      <exemption>trusted_merchant</exemption>
      <visa_merchant_id>00000000</visa_merchant_id>
    </sca_params>
  </payment_transaction>'
```

Master Synchronous 3 D Sv1 Attempted Authentication Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Sale3D');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('5555555555559997')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')


        // MPI Synchronous 3DSv1
        ->setMpiEci('01')
        ->setMpiCavv('hlk/+AMJkDARYwAAA2SqCUQAAAA=')
        ->setMpiXid('UVg1QThJU1FVV1hWWVQxQkxZNU8=')
        ->setMpiProtocolVersion('1');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.Sale3DRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    Sale3DRequest request = new Sale3DRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("5555555555559997");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");


    // Mpi Synchronous V1
    request.setMpiParams("{"eci"=>"01", "cavv"=>"hlk/+AMJkDARYwAAA2SqCUQAAAA=", "xid"=>"UVg1QThJU1FVV1hWVQxQkxZNU8=", "protocol_version"=>"1"}");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.sale3d(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "5555555555559997",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "business_attributes": {
      "event_start_date": "16-08-2020",
      "event_end_date": "25-08-2020",
      "event_organizer_id": "20192375",
      "event_id": "1912"
    },
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    },
    "mpi_params": {
      "eci": "01",
      "cavv": "hlk/+AMJkDARYwAAA2SqCUQAAAA=",
      "xid": "UVg1QThJU1FVV1hWWVQxQkxZNU8=",
      "protocol_version": "1"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>sale3d</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>100</amount>
    <currency>USD</currency>
    <card_holder>Travis Pastrana</card_holder>
    <card_number>5555555555559997</card_number>
    <expiration_month>12</expiration_month>
    <expiration_year>2021</expiration_year>
    <cvv>834</cvv>
    <customer_email>travis@example.com</customer_email>
    <customer_phone>+1987987987987</customer_phone>
    <business_attributes>
      <event_start_date>16-08-2020</event_start_date>
      <event_end_date>25-08-2020</event_end_date>
      <event_organizer_id>20192375</event_organizer_id>
      <event_id>1912</event_id>
    </business_attributes>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>10178</zip_code>
      <city>Los Angeles</city>
      <state>CA</state>
      <country>US</country>
    </billing_address>
    <mpi_params>
      <eci>01</eci>
      <cavv>hlk/+AMJkDARYwAAA2SqCUQAAAA=</cavv>
      <xid>UVg1QThJU1FVV1hWWVQxQkxZNU8=</xid>
      <protocol_version>1</protocol_version>
    </mpi_params>
  </payment_transaction>'
```

Asynchronous 3 D Sv1 Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Sale3D');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('4200000000000000')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')
        ->setNotificationUrl('https://www.example.com/notification')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.Sale3DRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;
import java.net.MalformedURLException;
import java.net.URL;

public class GenesisExample {
  public static void main() throws MalformedURLException {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    Sale3DRequest request = new Sale3DRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("4200000000000000");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");
    request.setNotificationUrl(new URL("https://www.example.com/notification"));
    request.setReturnSuccessUrl(new URL("http://www.example.com/success"));
    request.setReturnFailureUrl(new URL("http://www.example.com/failure"));

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.sale3d(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "4200000000000000",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    },
    "notification_url": "https://www.example.com/notification",
    "return_success_url": "http://www.example.com/success",
    "return_failure_url": "http://www.example.com/failure"
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>sale3d</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>100</amount>
    <currency>USD</currency>
    <card_holder>Travis Pastrana</card_holder>
    <card_number>4200000000000000</card_number>
    <expiration_month>12</expiration_month>
    <expiration_year>2021</expiration_year>
    <cvv>834</cvv>
    <customer_email>travis@example.com</customer_email>
    <customer_phone>+1987987987987</customer_phone>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>10178</zip_code>
      <city>Los Angeles</city>
      <state>CA</state>
      <country>US</country>
    </billing_address>
    <notification_url>https://www.example.com/notification</notification_url>
    <return_success_url>http://www.example.com/success</return_success_url>
    <return_failure_url>http://www.example.com/failure</return_failure_url>
  </payment_transaction>'
```

Visa Synchronous 3 D Sv2 Fully Authenticated Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Sale3D');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('4012000000060085')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')


        // MPI Synchronous 3DSv2
        ->setMpiEci('05')
        ->setMpiCavv('MDAwMDAwMDAwMDAxMTA2Njk5NFg=')
        ->setMpiProtocolVersion('2')
        ->setMpiDirectoryServerId('f418cc30-a98a-0138-0f10-00505690f91e')


        // Sca Params
        ->setScaExemption('trusted_merchant')
        ->setScaVisaMerchantId('00000000');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.Sale3DRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    Sale3DRequest request = new Sale3DRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("4012000000060085");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");


    // Mpi Synchronous V2
    request.setMpiParams("{"eci"=>"05", "cavv"=>"MDAwMDAwMDAwMDAxMTA2Njk5NFg=", "protocol_version"=>"2", "directory_server_id"=>"f418cc30-a98a-0138-0f10-00505690f91e"}");


    // Sca Params
    request.setScaExemption("trusted_merchant");
    request.setScaVisaMerchantId("00000000");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.sale3d(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "4012000000060085",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "business_attributes": {
      "event_start_date": "16-08-2020",
      "event_end_date": "25-08-2020",
      "event_organizer_id": "20192375",
      "event_id": "1912"
    },
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    },
    "mpi_params": {
      "eci": "05",
      "cavv": "MDAwMDAwMDAwMDAxMTA2Njk5NFg=",
      "protocol_version": "2",
      "directory_server_id": "f418cc30-a98a-0138-0f10-00505690f91e"
    },
    "sca_params": {
      "exemption": "trusted_merchant",
      "visa_merchant_id": "00000000"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>sale3d</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>100</amount>
    <currency>USD</currency>
    <card_holder>Travis Pastrana</card_holder>
    <card_number>4012000000060085</card_number>
    <expiration_month>12</expiration_month>
    <expiration_year>2021</expiration_year>
    <cvv>834</cvv>
    <customer_email>travis@example.com</customer_email>
    <customer_phone>+1987987987987</customer_phone>
    <business_attributes>
      <event_start_date>16-08-2020</event_start_date>
      <event_end_date>25-08-2020</event_end_date>
      <event_organizer_id>20192375</event_organizer_id>
      <event_id>1912</event_id>
    </business_attributes>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>10178</zip_code>
      <city>Los Angeles</city>
      <state>CA</state>
      <country>US</country>
    </billing_address>
    <mpi_params>
      <eci>05</eci>
      <cavv>MDAwMDAwMDAwMDAxMTA2Njk5NFg=</cavv>
      <protocol_version>2</protocol_version>
      <directory_server_id>f418cc30-a98a-0138-0f10-00505690f91e</directory_server_id>
    </mpi_params>
    <sca_params>
      <exemption>trusted_merchant</exemption>
      <visa_merchant_id>00000000</visa_merchant_id>
    </sca_params>
  </payment_transaction>'
```

Master Synchronous 3 D Sv2 Fully Authenticated Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Sale3D');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('5555555555559997')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')


        // MPI Synchronous 3DSv2
        ->setMpiEci('02')
        ->setMpiCavv('kAOImea0C/2ta1NZl4Hhwslmomqj')
        ->setMpiProtocolVersion('2')
        ->setMpiDirectoryServerId('f41ca5a0-a98a-0138-0f10-00505690f91e');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.Sale3DRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    Sale3DRequest request = new Sale3DRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("5555555555559997");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");


    // Mpi Synchronous V2
    request.setMpiParams("{"eci"=>"02", "cavv"=>"kAOImea0C/2ta1NZl4Hhwslmomqj", "protocol_version"=>"2", "directory_server_id"=>"f41ca5a0-a98a-0138-0f10-00505690f91e"}");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.sale3d(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "5555555555559997",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "business_attributes": {
      "event_start_date": "16-08-2020",
      "event_end_date": "25-08-2020",
      "event_organizer_id": "20192375",
      "event_id": "1912"
    },
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    },
    "mpi_params": {
      "eci": "02",
      "cavv": "kAOImea0C/2ta1NZl4Hhwslmomqj",
      "protocol_version": "2",
      "directory_server_id": "f41ca5a0-a98a-0138-0f10-00505690f91e"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>sale3d</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>100</amount>
    <currency>USD</currency>
    <card_holder>Travis Pastrana</card_holder>
    <card_number>5555555555559997</card_number>
    <expiration_month>12</expiration_month>
    <expiration_year>2021</expiration_year>
    <cvv>834</cvv>
    <customer_email>travis@example.com</customer_email>
    <customer_phone>+1987987987987</customer_phone>
    <business_attributes>
      <event_start_date>16-08-2020</event_start_date>
      <event_end_date>25-08-2020</event_end_date>
      <event_organizer_id>20192375</event_organizer_id>
      <event_id>1912</event_id>
    </business_attributes>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>10178</zip_code>
      <city>Los Angeles</city>
      <state>CA</state>
      <country>US</country>
    </billing_address>
    <mpi_params>
      <eci>02</eci>
      <cavv>kAOImea0C/2ta1NZl4Hhwslmomqj</cavv>
      <protocol_version>2</protocol_version>
      <directory_server_id>f41ca5a0-a98a-0138-0f10-00505690f91e</directory_server_id>
    </mpi_params>
  </payment_transaction>'
```

Visa Synchronous 3 D Sv2 Attempted Authentication Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Sale3D');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('4012000000060085')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')


        // MPI Synchronous 3DSv2
        ->setMpiEci('06')
        ->setMpiCavv('MDAwMDAwMDAwMDAxMTA2Njk5NFg=')
        ->setMpiProtocolVersion('2')
        ->setMpiDirectoryServerId('f4205460-a98a-0138-0f10-00505690f91e')


        // Sca Params
        ->setScaExemption('trusted_merchant')
        ->setScaVisaMerchantId('00000000');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.Sale3DRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    Sale3DRequest request = new Sale3DRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("4012000000060085");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");


    // Mpi Synchronous V2
    request.setMpiParams("{"eci"=>"06", "cavv"=>"MDAwMDAwMDAwMDAxMTA2Njk5NFg=", "protocol_version"=>"2", "directory_server_id"=>"f4205460-a98a-0138-0f10-00505690f91e"}");


    // Sca Params
    request.setScaExemption("trusted_merchant");
    request.setScaVisaMerchantId("00000000");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.sale3d(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "4012000000060085",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "business_attributes": {
      "event_start_date": "16-08-2020",
      "event_end_date": "25-08-2020",
      "event_organizer_id": "20192375",
      "event_id": "1912"
    },
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    },
    "mpi_params": {
      "eci": "06",
      "cavv": "MDAwMDAwMDAwMDAxMTA2Njk5NFg=",
      "protocol_version": "2",
      "directory_server_id": "f4205460-a98a-0138-0f10-00505690f91e"
    },
    "sca_params": {
      "exemption": "trusted_merchant",
      "visa_merchant_id": "00000000"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>sale3d</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>100</amount>
    <currency>USD</currency>
    <card_holder>Travis Pastrana</card_holder>
    <card_number>4012000000060085</card_number>
    <expiration_month>12</expiration_month>
    <expiration_year>2021</expiration_year>
    <cvv>834</cvv>
    <customer_email>travis@example.com</customer_email>
    <customer_phone>+1987987987987</customer_phone>
    <business_attributes>
      <event_start_date>16-08-2020</event_start_date>
      <event_end_date>25-08-2020</event_end_date>
      <event_organizer_id>20192375</event_organizer_id>
      <event_id>1912</event_id>
    </business_attributes>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>10178</zip_code>
      <city>Los Angeles</city>
      <state>CA</state>
      <country>US</country>
    </billing_address>
    <mpi_params>
      <eci>06</eci>
      <cavv>MDAwMDAwMDAwMDAxMTA2Njk5NFg=</cavv>
      <protocol_version>2</protocol_version>
      <directory_server_id>f4205460-a98a-0138-0f10-00505690f91e</directory_server_id>
    </mpi_params>
    <sca_params>
      <exemption>trusted_merchant</exemption>
      <visa_merchant_id>00000000</visa_merchant_id>
    </sca_params>
  </payment_transaction>'
```

Master Synchronous 3 D Sv2 Attempted Authentication Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Sale3D');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('5555555555559997')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')


        // MPI Synchronous 3DSv2
        ->setMpiEci('01')
        ->setMpiCavv('kEOImea0C/2ta1NZl4Hhwslmomqj')
        ->setMpiProtocolVersion('2')
        ->setMpiDirectoryServerId('f4243f40-a98a-0138-0f10-00505690f91e');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.Sale3DRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    Sale3DRequest request = new Sale3DRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("5555555555559997");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");


    // Mpi Synchronous V2
    request.setMpiParams("{"eci"=>"01", "cavv"=>"kEOImea0C/2ta1NZl4Hhwslmomqj", "protocol_version"=>"2", "directory_server_id"=>"f4243f40-a98a-0138-0f10-00505690f91e"}");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.sale3d(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "5555555555559997",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "business_attributes": {
      "event_start_date": "16-08-2020",
      "event_end_date": "25-08-2020",
      "event_organizer_id": "20192375",
      "event_id": "1912"
    },
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    },
    "mpi_params": {
      "eci": "01",
      "cavv": "kEOImea0C/2ta1NZl4Hhwslmomqj",
      "protocol_version": "2",
      "directory_server_id": "f4243f40-a98a-0138-0f10-00505690f91e"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>sale3d</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>100</amount>
    <currency>USD</currency>
    <card_holder>Travis Pastrana</card_holder>
    <card_number>5555555555559997</card_number>
    <expiration_month>12</expiration_month>
    <expiration_year>2021</expiration_year>
    <cvv>834</cvv>
    <customer_email>travis@example.com</customer_email>
    <customer_phone>+1987987987987</customer_phone>
    <business_attributes>
      <event_start_date>16-08-2020</event_start_date>
      <event_end_date>25-08-2020</event_end_date>
      <event_organizer_id>20192375</event_organizer_id>
      <event_id>1912</event_id>
    </business_attributes>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>10178</zip_code>
      <city>Los Angeles</city>
      <state>CA</state>
      <country>US</country>
    </billing_address>
    <mpi_params>
      <eci>01</eci>
      <cavv>kEOImea0C/2ta1NZl4Hhwslmomqj</cavv>
      <protocol_version>2</protocol_version>
      <directory_server_id>f4243f40-a98a-0138-0f10-00505690f91e</directory_server_id>
    </mpi_params>
  </payment_transaction>'
```

Asynchronous 3 D Sv2 Frictionless Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Sale3D');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('4012000000060085')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')
        ->setNotificationUrl('https://www.example.com/notification')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')


        // Threeds V2 Params
        ->setThreedsControl('{"device_type"=>"browser", "challenge_window_size"=>"full_screen", "challenge_indicator"=>"preference"}')
        ->setThreedsPurchase('{"category"=>"service"}')
        ->setThreedsRecurring('{"expiration_date"=>"16-01-2021", "frequency"=>30}')
        ->setThreedsMerchantRisk('{"shipping_indicator"=>"verified_address", "delivery_timeframe"=>"electronic", "reorder_items_indicator"=>"reordered", "pre_order_purchase_indicator"=>"merchandise_available",
        ->setThreedsCardHolderAccount('{"creation_date"=>"16-07-2019", "update_indicator"=>"more_than_60days", "last_change_date"=>"16-04-2020", "password_change_indicator"=>"no_change", "password_change_date"
        ->setThreedsBrowser('{"accept_header"=>"Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser", "java_enabled"=>"true", "language"=>"en-GB", "color_depth"=>"
        ->setThreedsSdk('{"interface"=>"native", "ui_types"=>{"ui_type"=>"multi_select"}, "application_id"=>"fc1650c0-5778-0138-8205-2cbc32a32d65", "encrypted_data"=>"encrypted-data-here", "ephemeral_public_ke

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.Sale3DRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;
import java.net.MalformedURLException;
import java.net.URL;

public class GenesisExample {
  public static void main() throws MalformedURLException {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    Sale3DRequest request = new Sale3DRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("4012000000060085");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");
    request.setNotificationUrl(new URL("https://www.example.com/notification"));
    request.setReturnSuccessUrl(new URL("http://www.example.com/success"));
    request.setReturnFailureUrl(new URL("http://www.example.com/failure"));


    // Threeds V2 Params
    request.setThreedsControl("{"device_type"=>"browser", "challenge_window_size"=>"full_screen", "challenge_indicator"=>"preference"}");
    request.setThreedsPurchase("{"category"=>"service"}");
    request.setThreedsRecurring("{"expiration_date"=>"16-01-2021", "frequency"=>30}");
    request.setThreedsMerchantRisk("{"shipping_indicator"=>"verified_address", "delivery_timeframe"=>"electronic", "reorder_items_indicator"=>"reordered", "pre_order_purchase_indicator"=>"merchandise_available
    request.setThreedsCardHolderAccount("{"creation_date"=>"16-07-2019", "update_indicator"=>"more_than_60days", "last_change_date"=>"16-04-2020", "password_change_indicator"=>"no_change", "password_change_da
    request.setThreedsBrowser("{"accept_header"=>"Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser", "java_enabled"=>"true", "language"=>"en-GB", "color_depth"
    request.setThreedsSdk("{"interface"=>"native", "ui_types"=>{"ui_type"=>"multi_select"}, "application_id"=>"fc1650c0-5778-0138-8205-2cbc32a32d65", "encrypted_data"=>"encrypted-data-here", "ephemeral_public_

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.sale3d(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "4012000000060085",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "billing_address": {
```

```
      "billing_address": {
        "first_name": "Travis",
        "last_name": "Pastrana",
        "address1": "Muster Str. 12",
        "zip_code": "10178",
        "city": "Los Angeles",
        "state": "CA",
        "country": "US"
      },
      "notification_url": "https://www.example.com/notification",
      "return_success_url": "http://www.example.com/success",
      "return_failure_url": "http://www.example.com/failure",
      "threeds_v2_params": {
        "control": {
          "device_type": "browser",
          "challenge_window_size": "full_screen",
          "challenge_indicator": "preference"
        },
        "purchase": {
          "category": "service"
        },
        "recurring": {
          "expiration_date": "16-01-2021",
          "frequency": 30
        },
        "merchant_risk": {
          "shipping_indicator": "verified_address",
          "delivery_timeframe": "electronic",
          "reorder_items_indicator": "reordered",
          "pre_order_purchase_indicator": "merchandise_available",
          "pre_order_date": "16-08-2020",
          "gift_card": "true",
          "gift_card_count": "2"
        },
        "card_holder_account": {
          "creation_date": "16-07-2019",
          "update_indicator": "more_than_60days",
          "last_change_date": "16-04-2020",
          "password_change_indicator": "no_change",
          "password_change_date": "01-07-2020",
          "shipping_address_usage_indicator": "current_transaction",
          "shipping_address_date_first_used": "11-07-2020",
          "transactions_activity_last_24_hours": "2",
          "transactions_activity_previous_year": "10",
          "provision_attempts_last_24_hours": "1",
          "purchases_count_last_6_months": "5",
          "suspicious_activity_indicator": "no_suspicious_observed",
          "registration_indicator": "30_to_60_days",
          "registration_date": "16-07-2018"
        },
        "browser": {
          "accept_header": "Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser",
          "java_enabled": "true",
          "language": "en-GB",
          "color_depth": "32",
          "screen_height": "900",
          "screen_width": "1440",
          "time_zone_offset": "-120",
          "user_agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.97 Safari/537.36"
        },
        "sdk": {
          "interface": "native",
          "ui_types": {
            "ui_type": "multi_select"
          },
          "application_id": "fc1650c0-5778-0138-8205-2cbc32a32d65",
          "encrypted_data": "encrypted-data-here",
          "ephemeral_public_key_pair": "public-key-pair",
          "max_timeout": "10",
          "reference_number": "sdk-reference-number-here"
        }
      }
    }
  ).send()
    .then(success)
    .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>sale3d</transaction_type>
   <transaction_id>43671</transaction_id>
   <usage>40208 concert tickets</usage>
   <remote_ip>245.253.2.12</remote_ip>
   <amount>100</amount>
   <currency>USD</currency>
   <card_holder>Travis Pastrana</card_holder>
   <card_number>4012000000060085</card_number>
   <expiration_month>12</expiration_month>
   <expiration_year>2021</expiration_year>
   <cvv>834</cvv>
   <customer_email>travis@example.com</customer_email>
   <customer_phone>+1987987987987</customer_phone>
   <billing_address>
     <first_name>Travis</first_name>
     <last_name>Pastrana</last_name>
     <address1>Muster Str. 12</address1>
     <zip_code>10178</zip_code>
     <city>Los Angeles</city>
     <state>CA</state>
     <country>US</country>
   </billing_address>
   <notification_url>https://www.example.com/notification</notification_url>
   <return_success_url>http://www.example.com/success</return_success_url>
   <return_failure_url>http://www.example.com/failure</return_failure_url>
   <threeds_v2_params>
     <control>
       <device_type>browser</device_type>
       <challenge_window_size>full_screen</challenge_window_size>
       <challenge_indicator>preference</challenge_indicator>
     </control>
     <purchase>
       <category>service</category>
     </purchase>
     <merchant_risk>
       <shipping_indicator>verified_address</shipping_indicator>
       <delivery_timeframe>electronic</delivery_timeframe>
       <reorder_items_indicator>reordered</reorder_items_indicator>
       <pre_order_purchase_indicator>merchandise_available</pre_order_purchase_indicator>
       <pre_order_date>16-08-2020</pre_order_date>
       <gift_card>true</gift_card>
       <gift_card_count>2</gift_card_count>
     </merchant_risk>
     <card_holder_account>
       <creation_date>16-07-2019</creation_date>
       <update_indicator>more_than_60days</update_indicator>
       <last_change_date>16-04-2020</last_change_date>
       <password_change_indicator>no_change</password_change_indicator>
       <password_change_date>01-07-2020</password_change_date>
       <shipping_address_usage_indicator>current_transaction</shipping_address_usage_indicator>
       <shipping_address_date_first_used>11-07-2020</shipping_address_date_first_used>
       <transactions_activity_last_24_hours>2</transactions_activity_last_24_hours>
       <transactions_activity_previous_year>10</transactions_activity_previous_year>
       <provision_attempts_last_24_hours>1</provision_attempts_last_24_hours>
       <purchases_count_last_6_months>5</purchases_count_last_6_months>
       <suspicious_activity_indicator>no_suspicious_observed</suspicious_activity_indicator>
       <registration_indicator>30_to_60_days</registration_indicator>
       <registration_date>16-07-2018</registration_date>
     </card_holder_account>
     <browser>
       <accept_header>Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser</accept_header>
       <java_enabled>true</java_enabled>
       <language>en-GB</language>
       <color_depth>32</color_depth>
       <screen_height>900</screen_height>
       <screen_width>1440</screen_width>
       <time_zone_offset>-120</time_zone_offset>
       <user_agent>Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.97 Safari/537.36</user_agent>
     </browser>
     <sdk>
       <interface>native</interface>
       <ui_types>
         <ui_type>multi_select</ui_type>
       </ui_types>
       <application_id>fc1650c0-5778-0138-8205-2cbc32a32d65</application_id>
       <encrypted_data>encrypted-data-here</encrypted_data>
       <ephemeral_public_key_pair>public-key-pair</ephemeral_public_key_pair>
       <max_timeout>10</max_timeout>
       <reference_number>sdk-reference-number-here</reference_number>
     </sdk>
   </threeds_v2_params>
 </payment_transaction>'
```

Asynchronous 3 D Sv2 Frictionless With 3ds Method Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Sale3D');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('4066330000000004')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')
        ->setNotificationUrl('https://www.example.com/notification')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')


        // Threeds V2 Params
        ->setThreedsControl('{"device_type"=>"browser", "challenge_window_size"=>"full_screen", "challenge_indicator"=>"preference"}')
        ->setThreedsPurchase('{"category"=>"service"}')
        ->setThreedsRecurring('{"expiration_date"=>"16-01-2021", "frequency"=>30}')
        ->setThreedsMerchantRisk('{"shipping_indicator"=>"verified_address", "delivery_timeframe"=>"electronic", "reorder_items_indicator"=>"reordered", "pre_order_purchase_indicator"=>"merchandise_available",
        ->setThreedsCardHolderAccount('{"creation_date"=>"16-07-2019", "update_indicator"=>"more_than_60days", "last_change_date"=>"16-04-2020", "password_change_indicator"=>"no_change", "password_change_date"
        ->setThreedsBrowser('{"accept_header"=>"Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser", "java_enabled"=>"true", "language"=>"en-GB", "color_depth"=>"
        ->setThreedsSdk('{"interface"=>"native", "ui_types"=>{"ui_type"=>"multi_select"}, "application_id"=>"fc1650c0-5778-0138-8205-2cbc32a32d65", "encrypted_data"=>"encrypted-data-here", "ephemeral_public_ke

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.Sale3DRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;
import java.net.MalformedURLException;
import java.net.URL;

public class GenesisExample {
  public static void main() throws MalformedURLException {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    Sale3DRequest request = new Sale3DRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("4066330000000004");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");
    request.setNotificationUrl(new URL("https://www.example.com/notification"));
    request.setReturnSuccessUrl(new URL("http://www.example.com/success"));
    request.setReturnFailureUrl(new URL("http://www.example.com/failure"));


    // Threeds V2 Params
    request.setThreedsControl("{"device_type"=>"browser", "challenge_window_size"=>"full_screen", "challenge_indicator"=>"preference"}");
    request.setThreedsPurchase("{"category"=>"service"}");
    request.setThreedsRecurring("{"expiration_date"=>"16-01-2021", "frequency"=>30}");
    request.setThreedsMerchantRisk("{"shipping_indicator"=>"verified_address", "delivery_timeframe"=>"electronic", "reorder_items_indicator"=>"reordered", "pre_order_purchase_indicator"=>"merchandise_available
    request.setThreedsCardHolderAccount("{"creation_date"=>"16-07-2019", "update_indicator"=>"more_than_60days", "last_change_date"=>"16-04-2020", "password_change_indicator"=>"no_change", "password_change_da
    request.setThreedsBrowser("{"accept_header"=>"Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser", "java_enabled"=>"true", "language"=>"en-GB", "color_depth"
    request.setThreedsSdk("{"interface"=>"native", "ui_types"=>{"ui_type"=>"multi_select"}, "application_id"=>"fc1650c0-5778-0138-8205-2cbc32a32d65", "encrypted_data"=>"encrypted-data-here", "ephemeral_public_

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.sale3d(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "4066330000000004",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "billing_address": {
```

```
      "billing_address": {
        "first_name": "Travis",
        "last_name": "Pastrana",
        "address1": "Muster Str. 12",
        "zip_code": "10178",
        "city": "Los Angeles",
        "state": "CA",
        "country": "US"
      },
      "notification_url": "https://www.example.com/notification",
      "return_success_url": "http://www.example.com/success",
      "return_failure_url": "http://www.example.com/failure",
      "threeds_v2_params": {
        "control": {
          "device_type": "browser",
          "challenge_window_size": "full_screen",
          "challenge_indicator": "preference"
        },
        "purchase": {
          "category": "service"
        },
        "recurring": {
          "expiration_date": "16-01-2021",
          "frequency": 30
        },
        "merchant_risk": {
          "shipping_indicator": "verified_address",
          "delivery_timeframe": "electronic",
          "reorder_items_indicator": "reordered",
          "pre_order_purchase_indicator": "merchandise_available",
          "pre_order_date": "16-08-2020",
          "gift_card": "true",
          "gift_card_count": "2"
        },
        "card_holder_account": {
          "creation_date": "16-07-2019",
          "update_indicator": "more_than_60days",
          "last_change_date": "16-04-2020",
          "password_change_indicator": "no_change",
          "password_change_date": "01-07-2020",
          "shipping_address_usage_indicator": "current_transaction",
          "shipping_address_date_first_used": "11-07-2020",
          "transactions_activity_last_24_hours": "2",
          "transactions_activity_previous_year": "10",
          "provision_attempts_last_24_hours": "1",
          "purchases_count_last_6_months": "5",
          "suspicious_activity_indicator": "no_suspicious_observed",
          "registration_indicator": "30_to_60_days",
          "registration_date": "16-07-2018"
        },
        "browser": {
          "accept_header": "Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser",
          "java_enabled": "true",
          "language": "en-GB",
          "color_depth": "32",
          "screen_height": "900",
          "screen_width": "1440",
          "time_zone_offset": "-120",
          "user_agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.97 Safari/537.36"
        },
        "sdk": {
          "interface": "native",
          "ui_types": {
            "ui_type": "multi_select"
          },
          "application_id": "fc1650c0-5778-0138-8205-2cbc32a32d65",
          "encrypted_data": "encrypted-data-here",
          "ephemeral_public_key_pair": "public-key-pair",
          "max_timeout": "10",
          "reference_number": "sdk-reference-number-here"
        }
      }
    }
  ).send()
    .then(success)
    .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>sale3d</transaction_type>
   <transaction_id>43671</transaction_id>
   <usage>40208 concert tickets</usage>
   <remote_ip>245.253.2.12</remote_ip>
   <amount>100</amount>
   <currency>USD</currency>
   <card_holder>Travis Pastrana</card_holder>
   <card_number>4066330000000004</card_number>
   <expiration_month>12</expiration_month>
   <expiration_year>2021</expiration_year>
   <cvv>834</cvv>
   <customer_email>travis@example.com</customer_email>
   <customer_phone>+1987987987987</customer_phone>
   <billing_address>
     <first_name>Travis</first_name>
     <last_name>Pastrana</last_name>
     <address1>Muster Str. 12</address1>
     <zip_code>10178</zip_code>
     <city>Los Angeles</city>
     <state>CA</state>
     <country>US</country>
   </billing_address>
   <notification_url>https://www.example.com/notification</notification_url>
   <return_success_url>http://www.example.com/success</return_success_url>
   <return_failure_url>http://www.example.com/failure</return_failure_url>
   <threeds_v2_params>
     <control>
       <device_type>browser</device_type>
       <challenge_window_size>full_screen</challenge_window_size>
       <challenge_indicator>preference</challenge_indicator>
     </control>
     <purchase>
       <category>service</category>
     </purchase>
     <merchant_risk>
       <shipping_indicator>verified_address</shipping_indicator>
       <delivery_timeframe>electronic</delivery_timeframe>
       <reorder_items_indicator>reordered</reorder_items_indicator>
       <pre_order_purchase_indicator>merchandise_available</pre_order_purchase_indicator>
       <pre_order_date>16-08-2020</pre_order_date>
       <gift_card>true</gift_card>
       <gift_card_count>2</gift_card_count>
     </merchant_risk>
     <card_holder_account>
       <creation_date>16-07-2019</creation_date>
       <update_indicator>more_than_60days</update_indicator>
       <last_change_date>16-04-2020</last_change_date>
       <password_change_indicator>no_change</password_change_indicator>
       <password_change_date>01-07-2020</password_change_date>
       <shipping_address_usage_indicator>current_transaction</shipping_address_usage_indicator>
       <shipping_address_date_first_used>11-07-2020</shipping_address_date_first_used>
       <transactions_activity_last_24_hours>2</transactions_activity_last_24_hours>
       <transactions_activity_previous_year>10</transactions_activity_previous_year>
       <provision_attempts_last_24_hours>1</provision_attempts_last_24_hours>
       <purchases_count_last_6_months>5</purchases_count_last_6_months>
       <suspicious_activity_indicator>no_suspicious_observed</suspicious_activity_indicator>
       <registration_indicator>30_to_60_days</registration_indicator>
       <registration_date>16-07-2018</registration_date>
     </card_holder_account>
     <browser>
       <accept_header>Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser</accept_header>
       <java_enabled>true</java_enabled>
       <language>en-GB</language>
       <color_depth>32</color_depth>
       <screen_height>900</screen_height>
       <screen_width>1440</screen_width>
       <time_zone_offset>-120</time_zone_offset>
       <user_agent>Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.97 Safari/537.36</user_agent>
     </browser>
     <sdk>
       <interface>native</interface>
       <ui_types>
         <ui_type>multi_select</ui_type>
       </ui_types>
       <application_id>fc1650c0-5778-0138-8205-2cbc32a32d65</application_id>
       <encrypted_data>encrypted-data-here</encrypted_data>
       <ephemeral_public_key_pair>public-key-pair</ephemeral_public_key_pair>
       <max_timeout>10</max_timeout>
       <reference_number>sdk-reference-number-here</reference_number>
     </sdk>
   </threeds_v2_params>
 </payment_transaction>'
```

Asynchronous 3 D Sv2 Challenge Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Sale3D');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('4918190000000002')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')
        ->setNotificationUrl('https://www.example.com/notification')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')


        // Threeds V2 Params
        ->setThreedsControl('{"device_type"=>"browser", "challenge_window_size"=>"full_screen", "challenge_indicator"=>"preference"}')
        ->setThreedsPurchase('{"category"=>"service"}')
        ->setThreedsRecurring('{"expiration_date"=>"16-01-2021", "frequency"=>30}')
        ->setThreedsMerchantRisk('{"shipping_indicator"=>"verified_address", "delivery_timeframe"=>"electronic", "reorder_items_indicator"=>"reordered", "pre_order_purchase_indicator"=>"merchandise_available",
        ->setThreedsCardHolderAccount('{"creation_date"=>"16-07-2019", "update_indicator"=>"more_than_60days", "last_change_date"=>"16-04-2020", "password_change_indicator"=>"no_change", "password_change_date"
        ->setThreedsBrowser('{"accept_header"=>"Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser", "java_enabled"=>"true", "language"=>"en-GB", "color_depth"=>"
        ->setThreedsSdk('{"interface"=>"native", "ui_types"=>{"ui_type"=>"multi_select"}, "application_id"=>"fc1650c0-5778-0138-8205-2cbc32a32d65", "encrypted_data"=>"encrypted-data-here", "ephemeral_public_ke

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.Sale3DRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;
import java.net.MalformedURLException;
import java.net.URL;

public class GenesisExample {
  public static void main() throws MalformedURLException {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    Sale3DRequest request = new Sale3DRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("4918190000000002");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");
    request.setNotificationUrl(new URL("https://www.example.com/notification"));
    request.setReturnSuccessUrl(new URL("http://www.example.com/success"));
    request.setReturnFailureUrl(new URL("http://www.example.com/failure"));


    // Threeds V2 Params
    request.setThreedsControl("{"device_type"=>"browser", "challenge_window_size"=>"full_screen", "challenge_indicator"=>"preference"}");
    request.setThreedsPurchase("{"category"=>"service"}");
    request.setThreedsRecurring("{"expiration_date"=>"16-01-2021", "frequency"=>30}");
    request.setThreedsMerchantRisk("{"shipping_indicator"=>"verified_address", "delivery_timeframe"=>"electronic", "reorder_items_indicator"=>"reordered", "pre_order_purchase_indicator"=>"merchandise_available
    request.setThreedsCardHolderAccount("{"creation_date"=>"16-07-2019", "update_indicator"=>"more_than_60days", "last_change_date"=>"16-04-2020", "password_change_indicator"=>"no_change", "password_change_da
    request.setThreedsBrowser("{"accept_header"=>"Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser", "java_enabled"=>"true", "language"=>"en-GB", "color_depth"
    request.setThreedsSdk("{"interface"=>"native", "ui_types"=>{"ui_type"=>"multi_select"}, "application_id"=>"fc1650c0-5778-0138-8205-2cbc32a32d65", "encrypted_data"=>"encrypted-data-here", "ephemeral_public_

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.sale3d(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "4918190000000002",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "billing_address": {
```

```
      "billing_address": {
        "first_name": "Travis",
        "last_name": "Pastrana",
        "address1": "Muster Str. 12",
        "zip_code": "10178",
        "city": "Los Angeles",
        "state": "CA",
        "country": "US"
      },
      "notification_url": "https://www.example.com/notification",
      "return_success_url": "http://www.example.com/success",
      "return_failure_url": "http://www.example.com/failure",
      "threeds_v2_params": {
        "control": {
          "device_type": "browser",
          "challenge_window_size": "full_screen",
          "challenge_indicator": "preference"
        },
        "purchase": {
          "category": "service"
        },
        "recurring": {
          "expiration_date": "16-01-2021",
          "frequency": 30
        },
        "merchant_risk": {
          "shipping_indicator": "verified_address",
          "delivery_timeframe": "electronic",
          "reorder_items_indicator": "reordered",
          "pre_order_purchase_indicator": "merchandise_available",
          "pre_order_date": "16-08-2020",
          "gift_card": "true",
          "gift_card_count": "2"
        },
        "card_holder_account": {
          "creation_date": "16-07-2019",
          "update_indicator": "more_than_60days",
          "last_change_date": "16-04-2020",
          "password_change_indicator": "no_change",
          "password_change_date": "01-07-2020",
          "shipping_address_usage_indicator": "current_transaction",
          "shipping_address_date_first_used": "11-07-2020",
          "transactions_activity_last_24_hours": "2",
          "transactions_activity_previous_year": "10",
          "provision_attempts_last_24_hours": "1",
          "purchases_count_last_6_months": "5",
          "suspicious_activity_indicator": "no_suspicious_observed",
          "registration_indicator": "30_to_60_days",
          "registration_date": "16-07-2018"
        },
        "browser": {
          "accept_header": "Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser",
          "java_enabled": "true",
          "language": "en-GB",
          "color_depth": "32",
          "screen_height": "900",
          "screen_width": "1440",
          "time_zone_offset": "-120",
          "user_agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.97 Safari/537.36"
        },
        "sdk": {
          "interface": "native",
          "ui_types": {
            "ui_type": "multi_select"
          },
          "application_id": "fc1650c0-5778-0138-8205-2cbc32a32d65",
          "encrypted_data": "encrypted-data-here",
          "ephemeral_public_key_pair": "public-key-pair",
          "max_timeout": "10",
          "reference_number": "sdk-reference-number-here"
        }
      }
    }
  ).send()
    .then(success)
    .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>sale3d</transaction_type>
   <transaction_id>43671</transaction_id>
   <usage>40208 concert tickets</usage>
   <remote_ip>245.253.2.12</remote_ip>
   <amount>100</amount>
   <currency>USD</currency>
   <card_holder>Travis Pastrana</card_holder>
   <card_number>4918190000000002</card_number>
   <expiration_month>12</expiration_month>
   <expiration_year>2021</expiration_year>
   <cvv>834</cvv>
   <customer_email>travis@example.com</customer_email>
   <customer_phone>+1987987987987</customer_phone>
   <billing_address>
     <first_name>Travis</first_name>
     <last_name>Pastrana</last_name>
     <address1>Muster Str. 12</address1>
     <zip_code>10178</zip_code>
     <city>Los Angeles</city>
     <state>CA</state>
     <country>US</country>
   </billing_address>
   <notification_url>https://www.example.com/notification</notification_url>
   <return_success_url>http://www.example.com/success</return_success_url>
   <return_failure_url>http://www.example.com/failure</return_failure_url>
   <threeds_v2_params>
     <control>
       <device_type>browser</device_type>
       <challenge_window_size>full_screen</challenge_window_size>
       <challenge_indicator>preference</challenge_indicator>
     </control>
     <purchase>
       <category>service</category>
     </purchase>
     <merchant_risk>
       <shipping_indicator>verified_address</shipping_indicator>
       <delivery_timeframe>electronic</delivery_timeframe>
       <reorder_items_indicator>reordered</reorder_items_indicator>
       <pre_order_purchase_indicator>merchandise_available</pre_order_purchase_indicator>
       <pre_order_date>16-08-2020</pre_order_date>
       <gift_card>true</gift_card>
       <gift_card_count>2</gift_card_count>
     </merchant_risk>
     <card_holder_account>
       <creation_date>16-07-2019</creation_date>
       <update_indicator>more_than_60days</update_indicator>
       <last_change_date>16-04-2020</last_change_date>
       <password_change_indicator>no_change</password_change_indicator>
       <password_change_date>01-07-2020</password_change_date>
       <shipping_address_usage_indicator>current_transaction</shipping_address_usage_indicator>
       <shipping_address_date_first_used>11-07-2020</shipping_address_date_first_used>
       <transactions_activity_last_24_hours>2</transactions_activity_last_24_hours>
       <transactions_activity_previous_year>10</transactions_activity_previous_year>
       <provision_attempts_last_24_hours>1</provision_attempts_last_24_hours>
       <purchases_count_last_6_months>5</purchases_count_last_6_months>
       <suspicious_activity_indicator>no_suspicious_observed</suspicious_activity_indicator>
       <registration_indicator>30_to_60_days</registration_indicator>
       <registration_date>16-07-2018</registration_date>
     </card_holder_account>
     <browser>
       <accept_header>Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser</accept_header>
       <java_enabled>true</java_enabled>
       <language>en-GB</language>
       <color_depth>32</color_depth>
       <screen_height>900</screen_height>
       <screen_width>1440</screen_width>
       <time_zone_offset>-120</time_zone_offset>
       <user_agent>Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.97 Safari/537.36</user_agent>
     </browser>
     <sdk>
       <interface>native</interface>
       <ui_types>
         <ui_type>multi_select</ui_type>
       </ui_types>
       <application_id>fc1650c0-5778-0138-8205-2cbc32a32d65</application_id>
       <encrypted_data>encrypted-data-here</encrypted_data>
       <ephemeral_public_key_pair>public-key-pair</ephemeral_public_key_pair>
       <max_timeout>10</max_timeout>
       <reference_number>sdk-reference-number-here</reference_number>
     </sdk>
   </threeds_v2_params>
 </payment_transaction>'
```

Asynchronous 3 D Sv2 Challenge With 3ds Method Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Sale3D');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('4938730000000001')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')
        ->setNotificationUrl('https://www.example.com/notification')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')


        // Threeds V2 Params
        ->setThreedsControl('{"device_type"=>"browser", "challenge_window_size"=>"full_screen", "challenge_indicator"=>"preference"}')
        ->setThreedsPurchase('{"category"=>"service"}')
        ->setThreedsRecurring('{"expiration_date"=>"16-01-2021", "frequency"=>30}')
        ->setThreedsMerchantRisk('{"shipping_indicator"=>"verified_address", "delivery_timeframe"=>"electronic", "reorder_items_indicator"=>"reordered", "pre_order_purchase_indicator"=>"merchandise_available",
        ->setThreedsCardHolderAccount('{"creation_date"=>"16-07-2019", "update_indicator"=>"more_than_60days", "last_change_date"=>"16-04-2020", "password_change_indicator"=>"no_change", "password_change_date"
        ->setThreedsBrowser('{"accept_header"=>"Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser", "java_enabled"=>"true", "language"=>"en-GB", "color_depth"=>"
        ->setThreedsSdk('{"interface"=>"native", "ui_types"=>{"ui_type"=>"multi_select"}, "application_id"=>"fc1650c0-5778-0138-8205-2cbc32a32d65", "encrypted_data"=>"encrypted-data-here", "ephemeral_public_ke

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.Sale3DRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;
import java.net.MalformedURLException;
import java.net.URL;

public class GenesisExample {
  public static void main() throws MalformedURLException {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    Sale3DRequest request = new Sale3DRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("4938730000000001");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");
    request.setNotificationUrl(new URL("https://www.example.com/notification"));
    request.setReturnSuccessUrl(new URL("http://www.example.com/success"));
    request.setReturnFailureUrl(new URL("http://www.example.com/failure"));


    // Threeds V2 Params
    request.setThreedsControl("{"device_type"=>"browser", "challenge_window_size"=>"full_screen", "challenge_indicator"=>"preference"}");
    request.setThreedsPurchase("{"category"=>"service"}");
    request.setThreedsRecurring("{"expiration_date"=>"16-01-2021", "frequency"=>30}");
    request.setThreedsMerchantRisk("{"shipping_indicator"=>"verified_address", "delivery_timeframe"=>"electronic", "reorder_items_indicator"=>"reordered", "pre_order_purchase_indicator"=>"merchandise_available
    request.setThreedsCardHolderAccount("{"creation_date"=>"16-07-2019", "update_indicator"=>"more_than_60days", "last_change_date"=>"16-04-2020", "password_change_indicator"=>"no_change", "password_change_da
    request.setThreedsBrowser("{"accept_header"=>"Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser", "java_enabled"=>"true", "language"=>"en-GB", "color_depth"
    request.setThreedsSdk("{"interface"=>"native", "ui_types"=>{"ui_type"=>"multi_select"}, "application_id"=>"fc1650c0-5778-0138-8205-2cbc32a32d65", "encrypted_data"=>"encrypted-data-here", "ephemeral_public_

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.sale3d(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "4938730000000001",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "billing_address": {
```

```
      "billing_address": {
        "first_name": "Travis",
        "last_name": "Pastrana",
        "address1": "Muster Str. 12",
        "zip_code": "10178",
        "city": "Los Angeles",
        "state": "CA",
        "country": "US"
      },
      "notification_url": "https://www.example.com/notification",
      "return_success_url": "http://www.example.com/success",
      "return_failure_url": "http://www.example.com/failure",
      "threeds_v2_params": {
        "control": {
          "device_type": "browser",
          "challenge_window_size": "full_screen",
          "challenge_indicator": "preference"
        },
        "purchase": {
          "category": "service"
        },
        "recurring": {
          "expiration_date": "16-01-2021",
          "frequency": 30
        },
        "merchant_risk": {
          "shipping_indicator": "verified_address",
          "delivery_timeframe": "electronic",
          "reorder_items_indicator": "reordered",
          "pre_order_purchase_indicator": "merchandise_available",
          "pre_order_date": "16-08-2020",
          "gift_card": "true",
          "gift_card_count": "2"
        },
        "card_holder_account": {
          "creation_date": "16-07-2019",
          "update_indicator": "more_than_60days",
          "last_change_date": "16-04-2020",
          "password_change_indicator": "no_change",
          "password_change_date": "01-07-2020",
          "shipping_address_usage_indicator": "current_transaction",
          "shipping_address_date_first_used": "11-07-2020",
          "transactions_activity_last_24_hours": "2",
          "transactions_activity_previous_year": "10",
          "provision_attempts_last_24_hours": "1",
          "purchases_count_last_6_months": "5",
          "suspicious_activity_indicator": "no_suspicious_observed",
          "registration_indicator": "30_to_60_days",
          "registration_date": "16-07-2018"
        },
        "browser": {
          "accept_header": "Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser",
          "java_enabled": "true",
          "language": "en-GB",
          "color_depth": "32",
          "screen_height": "900",
          "screen_width": "1440",
          "time_zone_offset": "-120",
          "user_agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.97 Safari/537.36"
        },
        "sdk": {
          "interface": "native",
          "ui_types": {
            "ui_type": "multi_select"
          },
          "application_id": "fc1650c0-5778-0138-8205-2cbc32a32d65",
          "encrypted_data": "encrypted-data-here",
          "ephemeral_public_key_pair": "public-key-pair",
          "max_timeout": "10",
          "reference_number": "sdk-reference-number-here"
        }
      }
    }
  ).send()
    .then(success)
    .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>sale3d</transaction_type>
   <transaction_id>43671</transaction_id>
   <usage>40208 concert tickets</usage>
   <remote_ip>245.253.2.12</remote_ip>
   <amount>100</amount>
   <currency>USD</currency>
   <card_holder>Travis Pastrana</card_holder>
   <card_number>4938730000000001</card_number>
   <expiration_month>12</expiration_month>
   <expiration_year>2021</expiration_year>
   <cvv>834</cvv>
   <customer_email>travis@example.com</customer_email>
   <customer_phone>+1987987987987</customer_phone>
   <billing_address>
     <first_name>Travis</first_name>
     <last_name>Pastrana</last_name>
     <address1>Muster Str. 12</address1>
     <zip_code>10178</zip_code>
     <city>Los Angeles</city>
     <state>CA</state>
     <country>US</country>
   </billing_address>
   <notification_url>https://www.example.com/notification</notification_url>
   <return_success_url>http://www.example.com/success</return_success_url>
   <return_failure_url>http://www.example.com/failure</return_failure_url>
   <threeds_v2_params>
     <control>
       <device_type>browser</device_type>
       <challenge_window_size>full_screen</challenge_window_size>
       <challenge_indicator>preference</challenge_indicator>
     </control>
     <purchase>
       <category>service</category>
     </purchase>
     <merchant_risk>
       <shipping_indicator>verified_address</shipping_indicator>
       <delivery_timeframe>electronic</delivery_timeframe>
       <reorder_items_indicator>reordered</reorder_items_indicator>
       <pre_order_purchase_indicator>merchandise_available</pre_order_purchase_indicator>
       <pre_order_date>16-08-2020</pre_order_date>
       <gift_card>true</gift_card>
       <gift_card_count>2</gift_card_count>
     </merchant_risk>
     <card_holder_account>
       <creation_date>16-07-2019</creation_date>
       <update_indicator>more_than_60days</update_indicator>
       <last_change_date>16-04-2020</last_change_date>
       <password_change_indicator>no_change</password_change_indicator>
       <password_change_date>01-07-2020</password_change_date>
       <shipping_address_usage_indicator>current_transaction</shipping_address_usage_indicator>
       <shipping_address_date_first_used>11-07-2020</shipping_address_date_first_used>
       <transactions_activity_last_24_hours>2</transactions_activity_last_24_hours>
       <transactions_activity_previous_year>10</transactions_activity_previous_year>
       <provision_attempts_last_24_hours>1</provision_attempts_last_24_hours>
       <purchases_count_last_6_months>5</purchases_count_last_6_months>
       <suspicious_activity_indicator>no_suspicious_observed</suspicious_activity_indicator>
       <registration_indicator>30_to_60_days</registration_indicator>
       <registration_date>16-07-2018</registration_date>
     </card_holder_account>
     <browser>
       <accept_header>Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser</accept_header>
       <java_enabled>true</java_enabled>
       <language>en-GB</language>
       <color_depth>32</color_depth>
       <screen_height>900</screen_height>
       <screen_width>1440</screen_width>
       <time_zone_offset>-120</time_zone_offset>
       <user_agent>Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.97 Safari/537.36</user_agent>
     </browser>
     <sdk>
       <interface>native</interface>
       <ui_types>
         <ui_type>multi_select</ui_type>
       </ui_types>
       <application_id>fc1650c0-5778-0138-8205-2cbc32a32d65</application_id>
       <encrypted_data>encrypted-data-here</encrypted_data>
       <ephemeral_public_key_pair>public-key-pair</ephemeral_public_key_pair>
       <max_timeout>10</max_timeout>
       <reference_number>sdk-reference-number-here</reference_number>
     </sdk>
   </threeds_v2_params>
 </payment_transaction>'
```

Asynchronous 3 D Sv2 Fallback 3dsv1 Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Sale3D');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('4901170000000003')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')
        ->setNotificationUrl('https://www.example.com/notification')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')


        // Threeds V2 Params
        ->setThreedsControl('{"device_type"=>"browser", "challenge_window_size"=>"full_screen", "challenge_indicator"=>"preference"}')
        ->setThreedsPurchase('{"category"=>"service"}')
        ->setThreedsRecurring('{"expiration_date"=>"16-01-2021", "frequency"=>30}')
        ->setThreedsMerchantRisk('{"shipping_indicator"=>"verified_address", "delivery_timeframe"=>"electronic", "reorder_items_indicator"=>"reordered", "pre_order_purchase_indicator"=>"merchandise_available",
        ->setThreedsCardHolderAccount('{"creation_date"=>"16-07-2019", "update_indicator"=>"more_than_60days", "last_change_date"=>"16-04-2020", "password_change_indicator"=>"no_change", "password_change_date"
        ->setThreedsBrowser('{"accept_header"=>"Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser", "java_enabled"=>"true", "language"=>"en-GB", "color_depth"=>"
        ->setThreedsSdk('{"interface"=>"native", "ui_types"=>{"ui_type"=>"multi_select"}, "application_id"=>"fc1650c0-5778-0138-8205-2cbc32a32d65", "encrypted_data"=>"encrypted-data-here", "ephemeral_public_ke

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.Sale3DRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;
import java.net.MalformedURLException;
import java.net.URL;

public class GenesisExample {
  public static void main() throws MalformedURLException {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    Sale3DRequest request = new Sale3DRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("4901170000000003");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");
    request.setNotificationUrl(new URL("https://www.example.com/notification"));
    request.setReturnSuccessUrl(new URL("http://www.example.com/success"));
    request.setReturnFailureUrl(new URL("http://www.example.com/failure"));


    // Threeds V2 Params
    request.setThreedsControl("{"device_type"=>"browser", "challenge_window_size"=>"full_screen", "challenge_indicator"=>"preference"}");
    request.setThreedsPurchase("{"category"=>"service"}");
    request.setThreedsRecurring("{"expiration_date"=>"16-01-2021", "frequency"=>30}");
    request.setThreedsMerchantRisk("{"shipping_indicator"=>"verified_address", "delivery_timeframe"=>"electronic", "reorder_items_indicator"=>"reordered", "pre_order_purchase_indicator"=>"merchandise_available
    request.setThreedsCardHolderAccount("{"creation_date"=>"16-07-2019", "update_indicator"=>"more_than_60days", "last_change_date"=>"16-04-2020", "password_change_indicator"=>"no_change", "password_change_da
    request.setThreedsBrowser("{"accept_header"=>"Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser", "java_enabled"=>"true", "language"=>"en-GB", "color_depth"
    request.setThreedsSdk("{"interface"=>"native", "ui_types"=>{"ui_type"=>"multi_select"}, "application_id"=>"fc1650c0-5778-0138-8205-2cbc32a32d65", "encrypted_data"=>"encrypted-data-here", "ephemeral_public_

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.sale3d(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "4901170000000003",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "billing_address": {
```

```
      "billing_address": {
        "first_name": "Travis",
        "last_name": "Pastrana",
        "address1": "Muster Str. 12",
        "zip_code": "10178",
        "city": "Los Angeles",
        "state": "CA",
        "country": "US"
      },
      "notification_url": "https://www.example.com/notification",
      "return_success_url": "http://www.example.com/success",
      "return_failure_url": "http://www.example.com/failure",
      "threeds_v2_params": {
        "control": {
          "device_type": "browser",
          "challenge_window_size": "full_screen",
          "challenge_indicator": "preference"
        },
        "purchase": {
          "category": "service"
        },
        "recurring": {
          "expiration_date": "16-01-2021",
          "frequency": 30
        },
        "merchant_risk": {
          "shipping_indicator": "verified_address",
          "delivery_timeframe": "electronic",
          "reorder_items_indicator": "reordered",
          "pre_order_purchase_indicator": "merchandise_available",
          "pre_order_date": "16-08-2020",
          "gift_card": "true",
          "gift_card_count": "2"
        },
        "card_holder_account": {
          "creation_date": "16-07-2019",
          "update_indicator": "more_than_60days",
          "last_change_date": "16-04-2020",
          "password_change_indicator": "no_change",
          "password_change_date": "01-07-2020",
          "shipping_address_usage_indicator": "current_transaction",
          "shipping_address_date_first_used": "11-07-2020",
          "transactions_activity_last_24_hours": "2",
          "transactions_activity_previous_year": "10",
          "provision_attempts_last_24_hours": "1",
          "purchases_count_last_6_months": "5",
          "suspicious_activity_indicator": "no_suspicious_observed",
          "registration_indicator": "30_to_60_days",
          "registration_date": "16-07-2018"
        },
        "browser": {
          "accept_header": "Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser",
          "java_enabled": "true",
          "language": "en-GB",
          "color_depth": "32",
          "screen_height": "900",
          "screen_width": "1440",
          "time_zone_offset": "-120",
          "user_agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.97 Safari/537.36"
        },
        "sdk": {
          "interface": "native",
          "ui_types": {
            "ui_type": "multi_select"
          },
          "application_id": "fc1650c0-5778-0138-8205-2cbc32a32d65",
          "encrypted_data": "encrypted-data-here",
          "ephemeral_public_key_pair": "public-key-pair",
          "max_timeout": "10",
          "reference_number": "sdk-reference-number-here"
        }
      }
    }
  ).send()
    .then(success)
    .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>sale3d</transaction_type>
   <transaction_id>43671</transaction_id>
   <usage>40208 concert tickets</usage>
   <remote_ip>245.253.2.12</remote_ip>
   <amount>100</amount>
   <currency>USD</currency>
   <card_holder>Travis Pastrana</card_holder>
   <card_number>4901170000000003</card_number>
   <expiration_month>12</expiration_month>
   <expiration_year>2021</expiration_year>
   <cvv>834</cvv>
   <customer_email>travis@example.com</customer_email>
   <customer_phone>+1987987987987</customer_phone>
   <billing_address>
     <first_name>Travis</first_name>
     <last_name>Pastrana</last_name>
     <address1>Muster Str. 12</address1>
     <zip_code>10178</zip_code>
     <city>Los Angeles</city>
     <state>CA</state>
     <country>US</country>
   </billing_address>
   <notification_url>https://www.example.com/notification</notification_url>
   <return_success_url>http://www.example.com/success</return_success_url>
   <return_failure_url>http://www.example.com/failure</return_failure_url>
   <threeds_v2_params>
     <control>
       <device_type>browser</device_type>
       <challenge_window_size>full_screen</challenge_window_size>
       <challenge_indicator>preference</challenge_indicator>
     </control>
     <purchase>
       <category>service</category>
     </purchase>
     <merchant_risk>
       <shipping_indicator>verified_address</shipping_indicator>
       <delivery_timeframe>electronic</delivery_timeframe>
       <reorder_items_indicator>reordered</reorder_items_indicator>
       <pre_order_purchase_indicator>merchandise_available</pre_order_purchase_indicator>
       <pre_order_date>16-08-2020</pre_order_date>
       <gift_card>true</gift_card>
       <gift_card_count>2</gift_card_count>
     </merchant_risk>
     <card_holder_account>
       <creation_date>16-07-2019</creation_date>
       <update_indicator>more_than_60days</update_indicator>
       <last_change_date>16-04-2020</last_change_date>
       <password_change_indicator>no_change</password_change_indicator>
       <password_change_date>01-07-2020</password_change_date>
       <shipping_address_usage_indicator>current_transaction</shipping_address_usage_indicator>
       <shipping_address_date_first_used>11-07-2020</shipping_address_date_first_used>
       <transactions_activity_last_24_hours>2</transactions_activity_last_24_hours>
       <transactions_activity_previous_year>10</transactions_activity_previous_year>
       <provision_attempts_last_24_hours>1</provision_attempts_last_24_hours>
       <purchases_count_last_6_months>5</purchases_count_last_6_months>
       <suspicious_activity_indicator>no_suspicious_observed</suspicious_activity_indicator>
       <registration_indicator>30_to_60_days</registration_indicator>
       <registration_date>16-07-2018</registration_date>
     </card_holder_account>
     <browser>
       <accept_header>Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser</accept_header>
       <java_enabled>true</java_enabled>
       <language>en-GB</language>
       <color_depth>32</color_depth>
       <screen_height>900</screen_height>
       <screen_width>1440</screen_width>
       <time_zone_offset>-120</time_zone_offset>
       <user_agent>Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.97 Safari/537.36</user_agent>
     </browser>
     <sdk>
       <interface>native</interface>
       <ui_types>
         <ui_type>multi_select</ui_type>
       </ui_types>
       <application_id>fc1650c0-5778-0138-8205-2cbc32a32d65</application_id>
       <encrypted_data>encrypted-data-here</encrypted_data>
       <ephemeral_public_key_pair>public-key-pair</ephemeral_public_key_pair>
       <max_timeout>10</max_timeout>
       <reference_number>sdk-reference-number-here</reference_number>
     </sdk>
   </threeds_v2_params>
 </payment_transaction>'
```

Asynchronous 3 D Sv2 Fallback 3dsv1 With 3ds Method Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Sale3D');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('4901164281364345')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')
        ->setNotificationUrl('https://www.example.com/notification')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')


        // Threeds V2 Params
        ->setThreedsControl('{"device_type"=>"browser", "challenge_window_size"=>"full_screen", "challenge_indicator"=>"preference"}')
        ->setThreedsPurchase('{"category"=>"service"}')
        ->setThreedsRecurring('{"expiration_date"=>"16-01-2021", "frequency"=>30}')
        ->setThreedsMerchantRisk('{"shipping_indicator"=>"verified_address", "delivery_timeframe"=>"electronic", "reorder_items_indicator"=>"reordered", "pre_order_purchase_indicator"=>"merchandise_available",
        ->setThreedsCardHolderAccount('{"creation_date"=>"16-07-2019", "update_indicator"=>"more_than_60days", "last_change_date"=>"16-04-2020", "password_change_indicator"=>"no_change", "password_change_date"
        ->setThreedsBrowser('{"accept_header"=>"Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser", "java_enabled"=>"true", "language"=>"en-GB", "color_depth"=>"
        ->setThreedsSdk('{"interface"=>"native", "ui_types"=>{"ui_type"=>"multi_select"}, "application_id"=>"fc1650c0-5778-0138-8205-2cbc32a32d65", "encrypted_data"=>"encrypted-data-here", "ephemeral_public_ke

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.Sale3DRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;
import java.net.MalformedURLException;
import java.net.URL;

public class GenesisExample {
  public static void main() throws MalformedURLException {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    Sale3DRequest request = new Sale3DRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("4901164281364345");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");
    request.setNotificationUrl(new URL("https://www.example.com/notification"));
    request.setReturnSuccessUrl(new URL("http://www.example.com/success"));
    request.setReturnFailureUrl(new URL("http://www.example.com/failure"));


    // Threeds V2 Params
    request.setThreedsControl("{"device_type"=>"browser", "challenge_window_size"=>"full_screen", "challenge_indicator"=>"preference"}");
    request.setThreedsPurchase("{"category"=>"service"}");
    request.setThreedsRecurring("{"expiration_date"=>"16-01-2021", "frequency"=>30}");
    request.setThreedsMerchantRisk("{"shipping_indicator"=>"verified_address", "delivery_timeframe"=>"electronic", "reorder_items_indicator"=>"reordered", "pre_order_purchase_indicator"=>"merchandise_available
    request.setThreedsCardHolderAccount("{"creation_date"=>"16-07-2019", "update_indicator"=>"more_than_60days", "last_change_date"=>"16-04-2020", "password_change_indicator"=>"no_change", "password_change_da
    request.setThreedsBrowser("{"accept_header"=>"Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser", "java_enabled"=>"true", "language"=>"en-GB", "color_depth"
    request.setThreedsSdk("{"interface"=>"native", "ui_types"=>{"ui_type"=>"multi_select"}, "application_id"=>"fc1650c0-5778-0138-8205-2cbc32a32d65", "encrypted_data"=>"encrypted-data-here", "ephemeral_public_

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.sale3d(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "4901164281364345",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "billing_address": {
```

```
      "billing_address": {
        "first_name": "Travis",
        "last_name": "Pastrana",
        "address1": "Muster Str. 12",
        "zip_code": "10178",
        "city": "Los Angeles",
        "state": "CA",
        "country": "US"
      },
      "notification_url": "https://www.example.com/notification",
      "return_success_url": "http://www.example.com/success",
      "return_failure_url": "http://www.example.com/failure",
      "threeds_v2_params": {
        "control": {
          "device_type": "browser",
          "challenge_window_size": "full_screen",
          "challenge_indicator": "preference"
        },
        "purchase": {
          "category": "service"
        },
        "recurring": {
          "expiration_date": "16-01-2021",
          "frequency": 30
        },
        "merchant_risk": {
          "shipping_indicator": "verified_address",
          "delivery_timeframe": "electronic",
          "reorder_items_indicator": "reordered",
          "pre_order_purchase_indicator": "merchandise_available",
          "pre_order_date": "16-08-2020",
          "gift_card": "true",
          "gift_card_count": "2"
        },
        "card_holder_account": {
          "creation_date": "16-07-2019",
          "update_indicator": "more_than_60days",
          "last_change_date": "16-04-2020",
          "password_change_indicator": "no_change",
          "password_change_date": "01-07-2020",
          "shipping_address_usage_indicator": "current_transaction",
          "shipping_address_date_first_used": "11-07-2020",
          "transactions_activity_last_24_hours": "2",
          "transactions_activity_previous_year": "10",
          "provision_attempts_last_24_hours": "1",
          "purchases_count_last_6_months": "5",
          "suspicious_activity_indicator": "no_suspicious_observed",
          "registration_indicator": "30_to_60_days",
          "registration_date": "16-07-2018"
        },
        "browser": {
          "accept_header": "Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser",
          "java_enabled": "true",
          "language": "en-GB",
          "color_depth": "32",
          "screen_height": "900",
          "screen_width": "1440",
          "time_zone_offset": "-120",
          "user_agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.97 Safari/537.36"
        },
        "sdk": {
          "interface": "native",
          "ui_types": {
            "ui_type": "multi_select"
          },
          "application_id": "fc1650c0-5778-0138-8205-2cbc32a32d65",
          "encrypted_data": "encrypted-data-here",
          "ephemeral_public_key_pair": "public-key-pair",
          "max_timeout": "10",
          "reference_number": "sdk-reference-number-here"
        }
      }
    }
  ).send()
    .then(success)
    .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>sale3d</transaction_type>
   <transaction_id>43671</transaction_id>
   <usage>40208 concert tickets</usage>
   <remote_ip>245.253.2.12</remote_ip>
   <amount>100</amount>
   <currency>USD</currency>
   <card_holder>Travis Pastrana</card_holder>
   <card_number>4901164281364345</card_number>
   <expiration_month>12</expiration_month>
   <expiration_year>2021</expiration_year>
   <cvv>834</cvv>
   <customer_email>travis@example.com</customer_email>
   <customer_phone>+1987987987987</customer_phone>
   <billing_address>
     <first_name>Travis</first_name>
     <last_name>Pastrana</last_name>
     <address1>Muster Str. 12</address1>
     <zip_code>10178</zip_code>
     <city>Los Angeles</city>
     <state>CA</state>
     <country>US</country>
   </billing_address>
   <notification_url>https://www.example.com/notification</notification_url>
   <return_success_url>http://www.example.com/success</return_success_url>
   <return_failure_url>http://www.example.com/failure</return_failure_url>
   <threeds_v2_params>
     <control>
       <device_type>browser</device_type>
       <challenge_window_size>full_screen</challenge_window_size>
       <challenge_indicator>preference</challenge_indicator>
     </control>
     <purchase>
       <category>service</category>
     </purchase>
     <merchant_risk>
       <shipping_indicator>verified_address</shipping_indicator>
       <delivery_timeframe>electronic</delivery_timeframe>
       <reorder_items_indicator>reordered</reorder_items_indicator>
       <pre_order_purchase_indicator>merchandise_available</pre_order_purchase_indicator>
       <pre_order_date>16-08-2020</pre_order_date>
       <gift_card>true</gift_card>
       <gift_card_count>2</gift_card_count>
     </merchant_risk>
     <card_holder_account>
       <creation_date>16-07-2019</creation_date>
       <update_indicator>more_than_60days</update_indicator>
       <last_change_date>16-04-2020</last_change_date>
       <password_change_indicator>no_change</password_change_indicator>
       <password_change_date>01-07-2020</password_change_date>
       <shipping_address_usage_indicator>current_transaction</shipping_address_usage_indicator>
       <shipping_address_date_first_used>11-07-2020</shipping_address_date_first_used>
       <transactions_activity_last_24_hours>2</transactions_activity_last_24_hours>
       <transactions_activity_previous_year>10</transactions_activity_previous_year>
       <provision_attempts_last_24_hours>1</provision_attempts_last_24_hours>
       <purchases_count_last_6_months>5</purchases_count_last_6_months>
       <suspicious_activity_indicator>no_suspicious_observed</suspicious_activity_indicator>
       <registration_indicator>30_to_60_days</registration_indicator>
       <registration_date>16-07-2018</registration_date>
     </card_holder_account>
     <browser>
       <accept_header>Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser</accept_header>
       <java_enabled>true</java_enabled>
       <language>en-GB</language>
       <color_depth>32</color_depth>
       <screen_height>900</screen_height>
       <screen_width>1440</screen_width>
       <time_zone_offset>-120</time_zone_offset>
       <user_agent>Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.97 Safari/537.36</user_agent>
     </browser>
     <sdk>
       <interface>native</interface>
       <ui_types>
         <ui_type>multi_select</ui_type>
       </ui_types>
       <application_id>fc1650c0-5778-0138-8205-2cbc32a32d65</application_id>
       <encrypted_data>encrypted-data-here</encrypted_data>
       <ephemeral_public_key_pair>public-key-pair</ephemeral_public_key_pair>
       <max_timeout>10</max_timeout>
       <reference_number>sdk-reference-number-here</reference_number>
     </sdk>
   </threeds_v2_params>
 </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **sale3d** |

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | required* | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| gaming | optional | "true" | Signifies whether a gaming transaction is performed. Gaming transactions usually use MCC 7995. Contact tech-support@e-comprocessing.com for more details |
| moto | optional | "true" | Signifies whether a MOTO (mail order telephone order) transaction is performed. Contacttech-support@e-comprocessing.com for more details |
| crypto | optional | "true" | Signifies whether a purchase of crypto-currency transaction is performed. Must be populated when purchasing crypto-currency with a VISA card and MCC is 6051. Must be populated when purchasing crypto-currency with a MASTER or INTL MAESTRO card and MCC is one of 6051, 6211. Contact tech-support@e-comprocessing.com for more details |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| notification_url | required[1] | url | URL at merchant where gateway sends outcome of transaction. |
| return_success_url | required[1] | url | URL where customer is sent to after successful payment |
| return_failure_url | required[1] | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| card_holder | required | string(255) | Full name of customer as printed on credit card (first name and last name at least) |
| card_number | required | 13 to 16 digits | Complete cc number of customer |
| cvv | required | 3 to 4 digits | cvv of cc, requirement is based on terminal configuration |
| expiration_month | required | MM | Expiration month as printed on credit card |
| expiration_year | required | YYYY | Expiration year as printed on credit card |
| token | optional | string(36) | See Tokenization for more details. If present, the cardholder parameters can be omitted. Cannot be set together with `remember_card` |
| remember_card | optional | "true" | See Tokenize. Tokenizes cardholder parameters. Cannot be set together with `token` |
| consumer_id | optional | string(10) | See Consumers and Tokenization. Combine with `remember_card` to tokenize or with `token` to use token |
| **credential_on_file** | required* | | See Credential On File (COF) for more details |
| initial_customer_initiated | required* | string(18) | Initial transaction initiated by customer. Required for external tokenization, and optional for gateway-based tokenization |
| subsequent_customer_initiated | required* | string(18) | Subsequent transaction initiated by customer. Required for external tokenization, and optional for gateway-based tokenization |
| merchant_unscheduled | optional | string(20) | Transaction is initiated by the merchant |
| customer_email | required* | e-mail address | Must contain valid e-mail of customer |
| customer_phone | required* | string(32) | Must contain valid phone number of customer |
| birth_date | required* | dd-mm-yyyy | Required when MCC is a Financial Services one (e.g. MCC 6012) and either card brand is Visa or Mastercard/Maestro with UK-based merchant, UK-based bin (domestic), and DEBIT card type |
| document_id | required* | sring(20) | Document ID of the consumer. See Document ID Parameter for more details. |
| fx_rate_id | optional | integer | See Get rates for FX Service. Offers the option to use a specific FX rate to convert the transaction processing amount. Used FX rate should have the same source currency as the processing currency. Contact tech-support@e-comprocessing.com for more details |
| **business_attributes** | required* | | Check business attributes section. |
| event_start_date | required* | dd-mm-yyyy | The date when event starts in format dd-mm-yyyy |
| event_end_date | required* | dd-mm-yyyy | The date when event ends in format dd-mm-yyyy |
| event_organizer_id | required* | string | |
| event_id | required* | string | |
| date_of_order | required* | dd-mm-yyyy | The date when order was placed in format dd-mm-yyyy |
| delivery_date | required* | dd-mm-yyyy | Date of the expected delivery in format dd-mm-yyyy |
| name_of_the_supplier | required* | string | |

| Parameter | Required | Format | Description |
|---|---|---|---|
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required* | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |
| **mpi_params** | required[2] | | |
| cavv | required[3] | string(255) | Verification Id of the authentication. Please note this can be the CAVV for Visa Card or UCAF to identify MasterCard. |
| eci | required[3] | string(255) | See Electronic Commerce Indicator as returned from the MPI for details |
| xid | required[3] | string(255) | Transaction ID generated by the 3DSv1 service that uniquely identifies the 3DSecure request. Note, it should not be submitted for transactions, which are though the 3DSv2 authentication protocol. |
| protocol_version | required[4] | string | The used 3DS protocol version. Default is 1 if not supplied. |
| directory_server_id | required[4] | string | The Directory Server ID used for 3DSecure transactions through the 3DSv2 authentication protocol. |
| **dynamic_descriptor_params** | optional | | |
| merchant_name | optional | string(25) | Allows to dynamically override the charge descriptor |
| merchant_city | optional | string(13) | Allows to dynamically override the merchant phone number |
| sub_merchant_id | optional | string(15) | Allows to dynamically override the sub-merchant ID. |
| **sca_params** | optional | | SCA params |
| exemption | optional | string | The exemption that the transaction should take advantage of. |
| visa_merchant_id | required[5] | string(8) | VMID assigned by Visa if participating in Trusted merchant program. |
| **threeds_v2_params** | required* | | 3DSv2 async parameters. They must be submitted in order to use the 3DSv2 authentication protocol in asynchronous workflow |
| **control** | required* | | General params for preferences in authentication flow and providing device interface information. |
| device_type | required* | string | Identifies the device channel of the consumer, **required** in the 3DSv2 authentication protocol. For more information, go to 3DSv2 control params |
| challenge_window_size | required* | string | Identifies the size of the challenge window for the consumer. For more information, go to3DSv2 control params |
| challenge_indicator | optional | string | The value has weight and might impact the decision whether a challenge will be required for the transaction or not. If not provided, it will be interpreted as **no_preference**. For more information, go to3DSv2 control params |
| **purchase** | required* | | Purchase related params providing with additional information regarding the order. |
| category | required* | string | Identifies the type of transaction being authenticated. This field is required in some markets.Accepted values are: **goods**, **service**, **check_acceptance**, **account_funding**, **quasi_cash**, **prepaid_activation**, **loan**. |
| **merchant_risk** | recommended | | Merchant risk assessment params. They are all optional, but recommended' |

| Parameter | Required | Format | Description |
|---|---|---|---|
| shipping_indicator | optional | string(16) | Indicator code that most accurately describes the shipping method for the cardholder specific transaction. If one or more items are included in the sale, use the Shipping Indicator code for the physical goods. If all digital goods, use the code that describes the most expensive item. Accepted values are: **same_as_billing**, **stored_address**, **verified_address**, **pick_up**, **digital_goods**, **travel**, **event_tickets**, **other**. |
| delivery_timeframe | optional | string(11) | Indicates the merchandise delivery timeframe. Accepted values are: **electronic**, **same_day**, **over_night**, **another_day**. |
| reorder_items_indicator | optional | string(10) | Indicates whether the cardholder is reordering previously purchased merchandise. Accepted values are: **first_time**, **reordered**. |
| pre_order_purchase_indicator | optional | string(21) | Indicates whether cardholder is placing an order for merchandise with a future-availability or release date. Accepted values are: **merchandise_available**, **future_availability**. |
| pre_order_date | optional | dd-mm-yyyy | For a pre-ordered purchase, the expected date that the merchandise will be available. |
| gift_card | optional | 'true' | Prepaid or gift card purchase. |
| gift_card_count | optional | integer | For prepaid or gift card purchase, total count of individual prepaid or gift cards/codes purchased. The value is limited to **99**. |
| **card_holder_account** | recommended | | Cardholder account additional information. They are all optional, but recommended, because they have a significant impact on approval rates |
| creation_date | optional | dd-mm-yyyy | Date that the cardholder opened the account with the 3DS Requester. |
| update_indicator | optional | string(19) | Length of time since the cardholder's account information with the 3DS Requestor was last changed. Includes Billing or Shipping address, new payment account, or new user(s) added. Accepted values are: **current_transaction**, **less_than_30days**, **30_to_60_days**, **more_than_60days**. |
| last_change_date | optional | dd-mm-yyyy | Date that the cardholder's account with the 3DS Requestor was last changed. Including Billing or Shipping address, new payment accout, or new user(s) added. |
| password_change_indicator | optional | string(18) | Length of time since the cardholder account with the 3DS Requestor had a password change or account reset. Accepted values are: **no_change**, **during_transaction**, **less_than_30days**, **30_to_60_days**, **more_than_60days**. |
| password_change_date | optional | dd-mm-yyyy | Date that cardholder's account with the 3DS Requestor had a password change or account reset. |
| shipping_address_usage_indicator | optional | string(19) | Indicates when the shipping address used for this transaction was first used with the 3DS Requestor. Accepted values are: **current_transaction**, **less_than_30days**, **30_to_60_days**, **more_than_60days**. |
| shipping_address_date_first_used | optional | dd-mm-yyyy | Date when the shipping address used for this transaction was first used with the 3DS Requestor. |
| transactions_activity_last_24_hours | optional | integer | Number of transactions (successful and abandoned) for this cardholder account with the 3DS Requestor across all payment accounts in the previous 24 hours. |
| transactions_activity_previous_year | optional | integer | Number of transactions (successful and abandoned) for this cardholder account with the 3DS Requestor across all payment accounts in the previous year. |
| provision_attempts_last_24_hours | optional | integer | Number of Add Card attempts in the last 24 hours. |
| purchases_count_last_6_months | optional | integer | Number of purchases with this cardholder account during the previous six months. |
| suspicious_activity_indicator | optional | string(22) | Indicates whether the 3DS Requestor has experienced suspicious activity (including previous fraud) on the cardholder account. Accepted values are: **no_suspicious_observed**, **suspicious_observed**. |
| registration_indicator | optional | string(19) | Indicates the length of time that the payment account was enrolled in the cardholder's account with the 3DS Requester. Accepted values are: **guest_checkout**, **current_transaction**, **less_than_30days**, **30_to_60_days**, **more_than_60days**. |
| registration_date | optional | dd-mm-yyyy | Date that the payment account was enrolled in the cardholder's account with the 3DS Requestor. |
| **browser** | required* | | For browser-based transactions. They are all *required* in case the **device_type** is set to **browser** |
| accept_header | required* | string(2048) | Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser. |
| java_enabled | required* | boolean | Boolean that represents the ability of the cardholder browser to execute Java. The value can be retrieved by accessing a property of the navigator with JavaScript, `navigator.javaEnabled`. |
| language | required* | string(8) | Value representing the browser language as defined in IETF BCP47. The value can be retrieved by accessing a property of the navigator with JavaScript, `navigator.language`. |
| color_depth | required* | integer | Value representing the bit depth of the colour palette for displaying images, in bits per pixel. Obtained from Cardholder browser using the `screen.colorDepth` property. The value can be one of **1**, **4**, **8**, **15**, **16**, **24**, **32**, **48** |
| screen_height | required* | integer | Total height of the Cardholder's screen in pixels. Value is returned from the screen.height property. |
| screen_width | required* | integer | Total width of the Cardholder's screen in pixels. Value is returned from the screen.width property. |
| time_zone_offset | required* | string(5) | Time difference between UTC time and the Cardholder browser local time, in minutes. The value can be retrieved using Javascript `getTimezoneOffset()` method. |
| user_agent | required* | string(2048) | Exact content of the HTTP user-agent header. |
| **sdk** | required* | | For application-based transactions. They are all *required* in case the **device_type** is set to **application** |

| Parameter | Required | Format | Description |
|---|---|---|---|
| interface | required* | string(6) | SDK Interface types that the device of the consumer supports for displaying specific challenge interfaces within the SDK. Accepted values are: **native**, **html**, **both**. |
| **ui_types** | required* | | Lists all UI types that the device of the consumer supports for displaying specific challenge interfaces within the SDK. |
| ui_type | required* | string(13) | UI type that the device of the consumer supports for displaying specific challenge interface. Accepted values are: **text**, **single_select**, **multi_select**, **out_of_bag**, **other_html**. |
| application_id | required* | string(36) | Universally unique ID created upon all installations and updates of the 3DS Requestor APp on a Customer Device. This will be newly generated and stored by the 3DS SDK for each installation or update. The field is limited to 36 characters and it shall have a canonical format as defined in IETF RFC 4122. |
| encrypted_data | required* | string(64000) | JWE Object as defined Section 6.2.2.1 containing data encrypted by the SDK for the DS to decrypt. The data will be present when sending to DS, but not present from DS to ACS. |
| ephemeral_public_key_pair | required* | string(256) | Public key component of the ephemeral key pair generated by the 3DS SDK and used to establish session keys between the 3DS SDK and ACS. In AReq, this data element is contained within the ACS Signed Content JWS Object. The field is limited to maximum 256 characters. |
| max_timeout | required* | integer | Indicates the maximum amount of time (in minutes) for all exchanges. The field shall have value greater or equals than 05. |
| reference_number | required* | string(32) | Identifies the vendor and version of the 3DS SDK that is integrated in a 3DS Requestor App, assigned by EMVCo when the 3DS SDK is approved. The field is limited to 32 characters. |

`required* = conditionally required`

`1 - required if mpi_params is not present, the transaction will be handled asynchronously. Not required if configured on Terminal or Merchant level. Contact [tech-support@#{email_domain_name}](mailto:tech-support@#{email_domain_name}) for more details.`

`2 - required if transaction should be handled synchronous.`

`3 - eci is always required if mpi_params is present. xid and cavv are not required for the 3D attempted only workflow, where the merchant does not have the xid/cavv, only the eci.`

`4 - protocol_version is only required if 3DSv2 is used. When missing 3DSv1 will be used. directory_server_id is mandatory when protocol_version is 2.`

`5 - visa_merchant_id is required when exemption value is trusted_merchant.`

Successful Synchronous Response

```
stdClass Object
(
    [transaction_type] => sale3d
    [status] => approved
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [consumer_id] => 123456
    [token] => ee946db8-d7db-4bb7-b608-b65b153e127d
    [avs_response_code] => 5I
    [avs_response_text] => Response provided by issuer processor; Address information not verified
    [authorization_code] => 345678
    [retrieval_reference_number] => 016813015184
    [response_code] => 00
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:16.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
    [partial_approval] => true
    [scheme_transaction_identifier] => 019091214161031
)
```

```xml
<payment_response content=[
    <transaction_type content=[sale3d]>
    <status content=[approved]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <consumer_id content=[123456]>
    <token content=[ee946db8-d7db-4bb7-b608-b65b153e127d]>
    <avs_response_code content=[5I]>
    <avs_response_text content=[Response provided by issuer processor; Address information not verified]>
    <authorization_code content=[345678]>
    <retrieval_reference_number content=[016813015184]>
    <response_code content=[00]>
    <timestamp content=[2020-07-15T12:08:16Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[true]>
    <partial_approval content=[true]>
    <scheme_transaction_identifier content=[019091214161031]>
]>
```

```json
{
    transaction_type: "sale3d",
    status: "approved",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    consumer_id: "123456",
    token: "ee946db8-d7db-4bb7-b608-b65b153e127d",
    avs_response_code: "5I",
    avs_response_text: "Response provided by issuer processor; Address information not verified",
    authorization_code: "345678",
    retrieval_reference_number: "016813015184",
    response_code: "00",
    timestamp: "2020-07-15T12:08:16Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "true",
    partial_approval: "true",
    scheme_transaction_identifier: "019091214161031",
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>sale3d</transaction_type>
  <status>approved</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <consumer_id>123456</consumer_id>
  <token>ee946db8-d7db-4bb7-b608-b65b153e127d</token>
  <avs_response_code>5I</avs_response_code>
  <avs_response_text>Response provided by issuer processor; Address information not verified</avs_response_text>
  <authorization_code>345678</authorization_code>
  <retrieval_reference_number>016813015184</retrieval_reference_number>
  <response_code>00</response_code>
  <timestamp>2020-07-15T12:08:16Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
  <partial_approval>true</partial_approval>
  <scheme_transaction_identifier>019091214161031</scheme_transaction_identifier>
</payment_transaction>
```


Successful Asynchronous Response

```
stdClass Object
(
    [transaction_type] => sale3d
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [consumer_id] => 123456
    [token] => ee946db8-d7db-4bb7-b608-b65b153e127d
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/threeds/authentication/fd3bbd0509d8ac3ef0708b76220b2509
    [redirect_url_type] => 3ds_v1_payer_authentication
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:16.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
    [scheme_transaction_identifier] => 019091214161031
)
```

```
<payment_response content=[
    <transaction_type content=[sale3d]>
    <status content=[pending_async]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <consumer_id content=[123456]>
    <token content=[ee946db8-d7db-4bb7-b608-b65b153e127d]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <redirect_url content=[https://staging.gate.e-comprocessing.net/threeds/authentication/fd3bbd0509d8ac3ef0708b76220b2509]>
    <redirect_url_type content=[3ds_v1_payer_authentication]>
    <timestamp content=[2020-07-15T12:08:16Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[true]>
    <scheme_transaction_identifier content=[019091214161031]>
]>
```

```
{
    transaction_type: "sale3d",
    status: "pending_async",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    consumer_id: "123456",
    token: "ee946db8-d7db-4bb7-b608-b65b153e127d",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    redirect_url: "https://staging.gate.e-comprocessing.net/threeds/authentication/fd3bbd0509d8ac3ef0708b76220b2509",
    redirect_url_type: "3ds_v1_payer_authentication",
    timestamp: "2020-07-15T12:08:16Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "true",
    scheme_transaction_identifier: "019091214161031",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>sale3d</transaction_type>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <consumer_id>123456</consumer_id>
  <token>ee946db8-d7db-4bb7-b608-b65b153e127d</token>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <redirect_url>https://staging.gate.e-comprocessing.net/threeds/authentication/fd3bbd0509d8ac3ef0708b76220b2509</redirect_url>
  <redirect_url_type>3ds_v1_payer_authentication</redirect_url_type>
  <timestamp>2020-07-15T12:08:16Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
  <scheme_transaction_identifier>019091214161031</scheme_transaction_identifier>
</payment_transaction>
```


Frictionless / Challenge With Synchronous 3ds Method Response

```
stdClass Object
(
    [transaction_type] => sale3d
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [consumer_id] => 123456
    [token] => ee946db8-d7db-4bb7-b608-b65b153e127d
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/threeds/threeds_method/1a096da019a4679a828baa175a306ae6
    [redirect_url_type] => 3ds_v2_threeds_method
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:16.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
    [scheme_transaction_identifier] => 019091214161031
    [authorization_code] => 345678
    [retrieval_reference_number] => 016813015184
)
```

```
<payment_response content=[
    <transaction_type content=[sale3d]>
    <status content=[pending_async]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <consumer_id content=[123456]>
    <token content=[ee946db8-d7db-4bb7-b608-b65b153e127d]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <redirect_url content=[https://staging.gate.e-comprocessing.net/threeds/threeds_method/1a096da019a4679a828baa175a306ae6]>
    <redirect_url_type content=[3ds_v2_threeds_method]>
    <timestamp content=[2020-07-15T12:08:16Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[true]>
    <scheme_transaction_identifier content=[019091214161031]>
    <authorization_code content=[345678]>
    <retrieval_reference_number content=[016813015184]>
]>
```

```
{
    transaction_type: "sale3d",
    status: "pending_async",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    consumer_id: "123456",
    token: "ee946db8-d7db-4bb7-b608-b65b153e127d",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    redirect_url: "https://staging.gate.e-comprocessing.net/threeds/threeds_method/1a096da019a4679a828baa175a306ae6",
    redirect_url_type: "3ds_v2_threeds_method",
    timestamp: "2020-07-15T12:08:16Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "true",
    scheme_transaction_identifier: "019091214161031",
    authorization_code: "345678",
    retrieval_reference_number: "016813015184",
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>sale3d</transaction_type>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <consumer_id>123456</consumer_id>
  <token>ee946db8-d7db-4bb7-b608-b65b153e127d</token>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <redirect_url>https://staging.gate.e-comprocessing.net/threeds/threeds_method/1a096da019a4679a828baa175a306ae6</redirect_url>
  <redirect_url_type>3ds_v2_threeds_method</redirect_url_type>
  <timestamp>2020-07-15T12:08:16Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
  <scheme_transaction_identifier>019091214161031</scheme_transaction_identifier>
  <authorization_code>345678</authorization_code>
  <retrieval_reference_number>016813015184</retrieval_reference_number>
</payment_transaction>
```


Frictionless / Challenge With Asynchronous 3ds Method Response


```
stdClass Object
(
    [transaction_type] => sale3d
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [consumer_id] => 123456
    [token] => ee946db8-d7db-4bb7-b608-b65b153e127d
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/threeds/threeds_method/1a096da019a4679a828baa175a306ae6/continue
    [redirect_url_type] => 3ds_v2_threeds_method
    [threeds_method_url] => https://staging.gate.e-comprocessing.net/threeds/threeds_method/649e1ff35c61
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:16.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
    [scheme_transaction_identifier] => 019091214161031
    [authorization_code] => 345678
    [retrieval_reference_number] => 016813015184
)
```



```
<payment_response content=[
    <transaction_type content=[sale3d]>
    <status content=[pending_async]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <consumer_id content=[123456]>
    <token content=[ee946db8-d7db-4bb7-b608-b65b153e127d]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <redirect_url content=[https://staging.gate.e-comprocessing.net/threeds/threeds_method/1a096da019a4679a828baa175a306ae6/continue]>
    <redirect_url_type content=[3ds_v2_threeds_method]>
    <threeds_method_url content=[https://staging.gate.e-comprocessing.net/threeds/threeds_method/649e1ff35c61]>
    <timestamp content=[2020-07-15T12:08:16Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[true]>
    <scheme_transaction_identifier content=[019091214161031]>
    <authorization_code content=[345678]>
    <retrieval_reference_number content=[016813015184]>
]>
```

```
{
    transaction_type: "sale3d",
    status: "pending_async",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    consumer_id: "123456",
    token: "ee946db8-d7db-4bb7-b608-b65b153e127d",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    redirect_url: "https://staging.gate.e-comprocessing.net/threeds/threeds_method/1a096da019a4679a828baa175a306ae6/continue",
    redirect_url_type: "3ds_v2_threeds_method",
    threeds_method_url: "https://staging.gate.e-comprocessing.net/threeds/threeds_method/649e1ff35c61",
    timestamp: "2020-07-15T12:08:16Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "true",
    scheme_transaction_identifier: "019091214161031",
    authorization_code: "345678",
    retrieval_reference_number: "016813015184",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>sale3d</transaction_type>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <consumer_id>123456</consumer_id>
  <token>ee946db8-d7db-4bb7-b608-b65b153e127d</token>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <redirect_url>https://staging.gate.e-comprocessing.net/threeds/threeds_method/1a096da019a4679a828baa175a306ae6/continue</redirect_url>
  <redirect_url_type>3ds_v2_threeds_method</redirect_url_type>
  <threeds_method_url>https://staging.gate.e-comprocessing.net/threeds/threeds_method/649e1ff35c61</threeds_method_url>
  <timestamp>2020-07-15T12:08:16Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
  <scheme_transaction_identifier>019091214161031</scheme_transaction_identifier>
  <authorization_code>345678</authorization_code>
  <retrieval_reference_number>016813015184</retrieval_reference_number>
</payment_transaction>
```


Challenge Without 3 Ds Method Response


```
stdClass Object
(
    [transaction_type] => sale3d
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [consumer_id] => 123456
    [token] => ee946db8-d7db-4bb7-b608-b65b153e127d
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/threeds/authentication/fd3bbd0509d8ac3ef0708b76220b2509
    [redirect_url_type] => 3ds_v2_challenge
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:16.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
    [scheme_transaction_identifier] => 019091214161031
)
```

```
<payment_response content=[
    <transaction_type content=[sale3d]>
    <status content=[pending_async]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <consumer_id content=[123456]>
    <token content=[ee946db8-d7db-4bb7-b608-b65b153e127d]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <redirect_url content=[https://staging.gate.e-comprocessing.net/threeds/authentication/fd3bbd0509d8ac3ef0708b76220b2509]>
    <redirect_url_type content=[3ds_v2_challenge]>
    <timestamp content=[2020-07-15T12:08:16Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[true]>
    <scheme_transaction_identifier content=[019091214161031]>
]>
```

```
{
    transaction_type: "sale3d",
    status: "pending_async",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    consumer_id: "123456",
    token: "ee946db8-d7db-4bb7-b608-b65b153e127d",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    redirect_url: "https://staging.gate.e-comprocessing.net/threeds/authentication/fd3bbd0509d8ac3ef0708b76220b2509",
    redirect_url_type: "3ds_v2_challenge",
    timestamp: "2020-07-15T12:08:16Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "true",
    scheme_transaction_identifier: "019091214161031",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>sale3d</transaction_type>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <consumer_id>123456</consumer_id>
  <token>ee946db8-d7db-4bb7-b608-b65b153e127d</token>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <redirect_url>https://staging.gate.e-comprocessing.net/threeds/authentication/fd3bbd0509d8ac3ef0708b76220b2509</redirect_url>
  <redirect_url_type>3ds_v2_challenge</redirect_url_type>
  <timestamp>2020-07-15T12:08:16Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
  <scheme_transaction_identifier>019091214161031</scheme_transaction_identifier>
</payment_transaction>
```


Fallback To 3ds1 Without 3ds Method Response


```
stdClass Object
(
    [transaction_type] => sale3d
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [consumer_id] => 123456
    [token] => ee946db8-d7db-4bb7-b608-b65b153e127d
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/threeds/authentication/fd3bbd0509d8ac3ef0708b76220b2509
    [redirect_url_type] => 3ds_v1_payer_authentication
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:16.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
    [scheme_transaction_identifier] => 019091214161031
)
```

```xml
<payment_response content=[
    <transaction_type content=[sale3d]>
    <status content=[pending_async]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <consumer_id content=[123456]>
    <token content=[ee946db8-d7db-4bb7-b608-b65b153e127d]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <redirect_url content=[https://staging.gate.e-comprocessing.net/threeds/authentication/fd3bbd0509d8ac3ef0708b76220b2509]>
    <redirect_url_type content=[3ds_v1_payer_authentication]>
    <timestamp content=[2020-07-15T12:08:16Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[true]>
    <scheme_transaction_identifier content=[019091214161031]>
]>
```

```json
{
    transaction_type: "sale3d",
    status: "pending_async",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    consumer_id: "123456",
    token: "ee946db8-d7db-4bb7-b608-b65b153e127d",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    redirect_url: "https://staging.gate.e-comprocessing.net/threeds/authentication/fd3bbd0509d8ac3ef0708b76220b2509",
    redirect_url_type: "3ds_v1_payer_authentication",
    timestamp: "2020-07-15T12:08:16Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "true",
    scheme_transaction_identifier: "019091214161031",
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>sale3d</transaction_type>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <consumer_id>123456</consumer_id>
  <token>ee946db8-d7db-4bb7-b608-b65b153e127d</token>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <redirect_url>https://staging.gate.e-comprocessing.net/threeds/authentication/fd3bbd0509d8ac3ef0708b76220b2509</redirect_url>
  <redirect_url_type>3ds_v1_payer_authentication</redirect_url_type>
  <timestamp>2020-07-15T12:08:16Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
  <scheme_transaction_identifier>019091214161031</scheme_transaction_identifier>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| consumer_id | string(10) | Consumer unique reference. See Consumers |
| token | string(36) | Plain-text token value. See Tokenization |
| gaming | string(255) | Signifies whether a gaming transaction is performed. Gaming transactions usually use MCC 7995, contact tech support for more details. |
| moto | string(255) | Signifies whether a MOTO (mail order telephone order) transaction is performed. Contact tech support for more details. |
| avs_response_code | string(255) | Generated by the card network on trying to match the billing address when performing the address verification. Optional, returned if config is enabled and acquirer supports it. Check AVS Status Codes for details. |
| avs_response_text | string(255) | Gives the human response text for the AVS response code above. Optional, returned if config is enabled and acquirer supports it. Check AVS Status Codes for details. |
| authorization_code | string(6) | Generated by the card network when an authorisation has occurred, used to identify that auth. Consists of 6 alphanumeric chars |
| retrieval_reference_number | string(255) | A reference number used for tracking all messages related to a given cardholder transaction returned by some acquirers. |

| Parameter | Type | Description |
|---|---|---|
| response_code | string(2) | Defines the issuer result of a transaction, the status of a message or some action taken or required. See issuer response codes for details |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| redirect_url | url | URL where the consumer has to be redirected to complete payment process unless a 3DSecure Method is required. For more information, to go 3DSv2 authentication flows |
| redirect_url_type | string(64) | The type of the redirect URL. It might be present only for asynchronous 3D transactions For more information, to go 3DSv2 authentication flows |
| threeds_method_url | url | 3DSecure Method URL. It will be present only then asynchronous 3DSecure Method is required for 3D transaction. For more information, to go 3DSv2 authentication flows |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08- 30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| partial_approval | string(4) | Optional, set to 'true' if partial approval happened. Partially approved amount is then in the amount field. Check Partial Approvals for details |
| sent_to_acquirer | string(255) | "true" or "false" |
| dynamic_descriptor_params | section | Optional, returned only if dynamic desc params are submitted on the API. Note here that the formatted dyn desc params are returned - as they would be submitted to the schemes for settlement. |
| scheme_transaction_identifier | string(32) | Id defined by card schemes |

Error Response

```
stdClass Object
(
    [transaction_type] => sale3d
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [response_code] => 57
    [code] => 340
    [technical_message] => expiration_year is invalid
    [message] => expiration_year is invalid
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:16.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => false
)
```

```
<payment_response content=[
    <transaction_type content=[sale3d]>
    <status content=[error]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <response_code content=[57]>
    <code content=[340]>
    <technical_message content=[expiration_year is invalid]>
    <message content=[expiration_year is invalid]>
    <timestamp content=[2020-07-15T12:08:16Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[false]>
]>
```

```
{
    transaction_type: "sale3d",
    status: "error",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    response_code: "57",
    code: "340",
    technical_message: "expiration_year is invalid",
    message: "expiration_year is invalid",
    timestamp: "2020-07-15T12:08:16Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "false",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>sale3d</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <response_code>57</response_code>
  <code>340</code>
  <technical_message>expiration_year is invalid</technical_message>
  <message>expiration_year is invalid</message>
  <timestamp>2020-07-15T12:08:16Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>false</sent_to_acquirer>
</payment_transaction>
```


**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| gaming | string(255) | Signifies whether a gaming transaction is performed. Gaming transactions usually use MCC 7995, contact tech support for more details. |
| moto | string(255) | Signifies whether a MOTO (mail order telephone order) transaction is performed. Contact tech support for more details. |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08- 30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| dynamic_descriptor_params | section | Optional, returned only if dynamic desc params are submitted on the API. Note here that the formatted dyn desc params are returned - as they would be submitted to the schemes for settlement. |

#### INIT RECURRING SALE 3D

InitRecurringSale3D transactions basically have the same request as standard InitRecurringSale transactions.

> InitRecurringSale3D transactions can be handled synchronous or asynchronous depending on the parameters passed. If mpi params is passed the workflow will be synchronous. If notification url, return success url and return failure url are passed the workflow will be asynchronous.

> This transaction type supports Tokenization.

> This transaction type supports Level 3 travel data.

> This transaction type could require business attributes.

Visa Synchronous 3 D Sv1 Fully Authenticated Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Recurring\InitRecurringSale3D');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('4711100000000000')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')


        // MPI Synchronous 3DSv1
        ->setMpiEci('05')
        ->setMpiCavv('MDAwMDAwMDAwMDAxMTA2Njk5NFg=')
        ->setMpiXid('0pv62FIrT5qQODB7DCewKgEBAQI=')
        ->setMpiProtocolVersion('1')


        // Sca Params
        ->setScaExemption('trusted_merchant')
        ->setScaVisaMerchantId('00000000');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.recurring.InitRecurringSale3D;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    InitRecurringSale3D request = new InitRecurringSale3D();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("4711100000000000");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");


    // Mpi Synchronous V1
    request.setMpiParams("{"eci"=>"05", "cavv"=>"MDAwMDAwMDAwMDAxMTA2Njk5NFg=", "xid"=>"0pv62FIrT5qQODB7DCewKgEBAQI=", "protocol_version"=>"1"}");


    // Sca Params
    request.setScaExemption("trusted_merchant");
    request.setScaVisaMerchantId("00000000");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.init_recurring_sale3d(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "4711100000000000",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "business_attributes": {
      "event_start_date": "16-08-2020",
      "event_end_date": "25-08-2020",
      "event_organizer_id": "20192375",
      "event_id": "1912"
    },
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    },
    "mpi_params": {
      "eci": "05",
      "cavv": "MDAwMDAwMDAwMDAxMTA2Njk5NFg=",
      "xid": "0pv62FIrT5qQODB7DCewKgEBAQI=",
      "protocol_version": "1"
    },
    "sca_params": {
      "exemption": "trusted_merchant",
      "visa_merchant_id": "00000000"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>init_recurring_sale3d</transaction_type>
   <transaction_id>43671</transaction_id>
   <usage>40208 concert tickets</usage>
   <remote_ip>245.253.2.12</remote_ip>
   <amount>100</amount>
   <currency>USD</currency>
   <card_holder>Travis Pastrana</card_holder>
   <card_number>4711100000000000</card_number>
   <expiration_month>12</expiration_month>
   <expiration_year>2021</expiration_year>
   <cvv>834</cvv>
   <customer_email>travis@example.com</customer_email>
   <customer_phone>+1987987987987</customer_phone>
   <business_attributes>
     <event_start_date>16-08-2020</event_start_date>
     <event_end_date>25-08-2020</event_end_date>
     <event_organizer_id>20192375</event_organizer_id>
     <event_id>1912</event_id>
   </business_attributes>
   <billing_address>
     <first_name>Travis</first_name>
     <last_name>Pastrana</last_name>
     <address1>Muster Str. 12</address1>
     <zip_code>10178</zip_code>
     <city>Los Angeles</city>
     <state>CA</state>
     <country>US</country>
   </billing_address>
   <mpi_params>
     <eci>05</eci>
     <cavv>MDAwMDAwMDAwMDAxMTA2Njk5NFg=</cavv>
     <xid>0pv62FIrT5qQODB7DCewKgEBAQI=</xid>
     <protocol_version>1</protocol_version>
   </mpi_params>
   <sca_params>
     <exemption>trusted_merchant</exemption>
     <visa_merchant_id>00000000</visa_merchant_id>
   </sca_params>
 </payment_transaction>'
```

Master Synchronous 3 D Sv1 Fully Authenticated Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Recurring\InitRecurringSale3D');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('5555555555559997')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')


        // MPI Synchronous 3DSv1
        ->setMpiEci('02')
        ->setMpiCavv('jlk/+AMJkDARYwAAA2SqCUQAAAA=')
        ->setMpiXid('UVg1QThJU1FVV1hWWVQxQkxZNU8=')
        ->setMpiProtocolVersion('1');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.recurring.InitRecurringSale3D;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    InitRecurringSale3D request = new InitRecurringSale3D();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("5555555555559997");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");


    // Mpi Synchronous V1
    request.setMpiParams("{"eci"=>"02", "cavv"=>"jlk/+AMJkDARYwAAA2SqCUQAAAA=", "xid"=>"UVg1QThJU1FVV1hWVQxQkxZNU8=", "protocol_version"=>"1"}");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.init_recurring_sale3d(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "5555555555559997",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "business_attributes": {
      "event_start_date": "16-08-2020",
      "event_end_date": "25-08-2020",
      "event_organizer_id": "20192375",
      "event_id": "1912"
    },
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    },
    "mpi_params": {
      "eci": "02",
      "cavv": "jlk/+AMJkDARYwAAA2SqCUQAAAA=",
      "xid": "UVg1QThJU1FVV1hWWVQxQkxZNU8=",
      "protocol_version": "1"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>init_recurring_sale3d</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>100</amount>
    <currency>USD</currency>
    <card_holder>Travis Pastrana</card_holder>
    <card_number>5555555555559997</card_number>
    <expiration_month>12</expiration_month>
    <expiration_year>2021</expiration_year>
    <cvv>834</cvv>
    <customer_email>travis@example.com</customer_email>
    <customer_phone>+1987987987987</customer_phone>
    <business_attributes>
      <event_start_date>16-08-2020</event_start_date>
      <event_end_date>25-08-2020</event_end_date>
      <event_organizer_id>20192375</event_organizer_id>
      <event_id>1912</event_id>
    </business_attributes>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>10178</zip_code>
      <city>Los Angeles</city>
      <state>CA</state>
      <country>US</country>
    </billing_address>
    <mpi_params>
      <eci>02</eci>
      <cavv>jlk/+AMJkDARYwAAA2SqCUQAAAA=</cavv>
      <xid>UVg1QThJU1FVV1hWWVQxQkxZNU8=</xid>
      <protocol_version>1</protocol_version>
    </mpi_params>
  </payment_transaction>'
```

Visa Synchronous 3 D Sv1 Attempted Authentication Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Recurring\InitRecurringSale3D');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('4711100000000000')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')


        // MPI Synchronous 3DSv1
        ->setMpiEci('06')
        ->setMpiCavv('MDAwMDAwMDAwMDAxMTA2Njk5NFg=')
        ->setMpiXid('0pv62FIrT5qQODB7DCewKgEBAQI=')
        ->setMpiProtocolVersion('1')


        // Sca Params
        ->setScaExemption('trusted_merchant')
        ->setScaVisaMerchantId('00000000');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.recurring.InitRecurringSale3D;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    InitRecurringSale3D request = new InitRecurringSale3D();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("4711100000000000");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");


    // Mpi Synchronous V1
    request.setMpiParams("{"eci"=>"06", "cavv"=>"MDAwMDAwMDAwMDAxMTA2Njk5NFg=", "xid"=>"0pv62FIrT5qQODB7DCewKgEBAQI=", "protocol_version"=>"1"}");


    // Sca Params
    request.setScaExemption("trusted_merchant");
    request.setScaVisaMerchantId("00000000");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.init_recurring_sale3d(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "4711100000000000",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "business_attributes": {
      "event_start_date": "16-08-2020",
      "event_end_date": "25-08-2020",
      "event_organizer_id": "20192375",
      "event_id": "1912"
    },
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    },
    "mpi_params": {
      "eci": "06",
      "cavv": "MDAwMDAwMDAwMDAxMTA2Njk5NFg=",
      "xid": "0pv62FIrT5qQODB7DCewKgEBAQI=",
      "protocol_version": "1"
    },
    "sca_params": {
      "exemption": "trusted_merchant",
      "visa_merchant_id": "00000000"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>init_recurring_sale3d</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>100</amount>
    <currency>USD</currency>
    <card_holder>Travis Pastrana</card_holder>
    <card_number>4711100000000000</card_number>
    <expiration_month>12</expiration_month>
    <expiration_year>2021</expiration_year>
    <cvv>834</cvv>
    <customer_email>travis@example.com</customer_email>
    <customer_phone>+1987987987987</customer_phone>
    <business_attributes>
      <event_start_date>16-08-2020</event_start_date>
      <event_end_date>25-08-2020</event_end_date>
      <event_organizer_id>20192375</event_organizer_id>
      <event_id>1912</event_id>
    </business_attributes>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>10178</zip_code>
      <city>Los Angeles</city>
      <state>CA</state>
      <country>US</country>
    </billing_address>
    <mpi_params>
      <eci>06</eci>
      <cavv>MDAwMDAwMDAwMDAxMTA2Njk5NFg=</cavv>
      <xid>0pv62FIrT5qQODB7DCewKgEBAQI=</xid>
      <protocol_version>1</protocol_version>
    </mpi_params>
    <sca_params>
      <exemption>trusted_merchant</exemption>
      <visa_merchant_id>00000000</visa_merchant_id>
    </sca_params>
  </payment_transaction>'
```

Master Synchronous 3 D Sv1 Attempted Authentication Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Recurring\InitRecurringSale3D');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('5555555555559997')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')


        // MPI Synchronous 3DSv1
        ->setMpiEci('01')
        ->setMpiCavv('hlk/+AMJkDARYwAAA2SqCUQAAAA=')
        ->setMpiXid('UVg1QThJU1FVV1hWWVQxQkxZNU8=')
        ->setMpiProtocolVersion('1');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.recurring.InitRecurringSale3D;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    InitRecurringSale3D request = new InitRecurringSale3D();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("5555555555559997");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");


    // Mpi Synchronous V1
    request.setMpiParams("{"eci"=>"01", "cavv"=>"hlk/+AMJkDARYwAAA2SqCUQAAAA=", "xid"=>"UVg1QThJU1FVV1hWVQxQkxZNU8=", "protocol_version"=>"1"}");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.init_recurring_sale3d(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "5555555555559997",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "business_attributes": {
      "event_start_date": "16-08-2020",
      "event_end_date": "25-08-2020",
      "event_organizer_id": "20192375",
      "event_id": "1912"
    },
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    },
    "mpi_params": {
      "eci": "01",
      "cavv": "hlk/+AMJkDARYwAAA2SqCUQAAAA=",
      "xid": "UVg1QThJU1FVV1hWWVQxQkxZNU8=",
      "protocol_version": "1"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>init_recurring_sale3d</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>100</amount>
    <currency>USD</currency>
    <card_holder>Travis Pastrana</card_holder>
    <card_number>5555555555559997</card_number>
    <expiration_month>12</expiration_month>
    <expiration_year>2021</expiration_year>
    <cvv>834</cvv>
    <customer_email>travis@example.com</customer_email>
    <customer_phone>+1987987987987</customer_phone>
    <business_attributes>
      <event_start_date>16-08-2020</event_start_date>
      <event_end_date>25-08-2020</event_end_date>
      <event_organizer_id>20192375</event_organizer_id>
      <event_id>1912</event_id>
    </business_attributes>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>10178</zip_code>
      <city>Los Angeles</city>
      <state>CA</state>
      <country>US</country>
    </billing_address>
    <mpi_params>
      <eci>01</eci>
      <cavv>hlk/+AMJkDARYwAAA2SqCUQAAAA=</cavv>
      <xid>UVg1QThJU1FVV1hWWVQxQkxZNU8=</xid>
      <protocol_version>1</protocol_version>
    </mpi_params>
  </payment_transaction>'
```

Asynchronous 3 D Sv1 Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Recurring\InitRecurringSale3D');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('4200000000000000')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')
        ->setNotificationUrl('https://www.example.com/notification')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.recurring.InitRecurringSale3D;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;
import java.net.MalformedURLException;
import java.net.URL;

public class GenesisExample {
  public static void main() throws MalformedURLException {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    InitRecurringSale3D request = new InitRecurringSale3D();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("4200000000000000");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");
    request.setNotificationUrl(new URL("https://www.example.com/notification"));
    request.setReturnSuccessUrl(new URL("http://www.example.com/success"));
    request.setReturnFailureUrl(new URL("http://www.example.com/failure"));

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.init_recurring_sale3d(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "4200000000000000",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    },
    "notification_url": "https://www.example.com/notification",
    "return_success_url": "http://www.example.com/success",
    "return_failure_url": "http://www.example.com/failure"
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>init_recurring_sale3d</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>100</amount>
    <currency>USD</currency>
    <card_holder>Travis Pastrana</card_holder>
    <card_number>4200000000000000</card_number>
    <expiration_month>12</expiration_month>
    <expiration_year>2021</expiration_year>
    <cvv>834</cvv>
    <customer_email>travis@example.com</customer_email>
    <customer_phone>+1987987987987</customer_phone>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>10178</zip_code>
      <city>Los Angeles</city>
      <state>CA</state>
      <country>US</country>
    </billing_address>
    <notification_url>https://www.example.com/notification</notification_url>
    <return_success_url>http://www.example.com/success</return_success_url>
    <return_failure_url>http://www.example.com/failure</return_failure_url>
  </payment_transaction>'
```

Visa Synchronous 3 D Sv2 Fully Authenticated Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Recurring\InitRecurringSale3D');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('4012000000060085')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')


        // MPI Synchronous 3DSv2
        ->setMpiEci('05')
        ->setMpiCavv('MDAwMDAwMDAwMDAxMTA2Njk5NFg=')
        ->setMpiProtocolVersion('2')
        ->setMpiDirectoryServerId('f47ccba0-a98a-0138-0f10-00505690f91e')


        // Sca Params
        ->setScaExemption('trusted_merchant')
        ->setScaVisaMerchantId('00000000');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.recurring.InitRecurringSale3D;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    InitRecurringSale3D request = new InitRecurringSale3D();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("4012000000060085");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");


    // Mpi Synchronous V2
    request.setMpiParams("{"eci"=>"05", "cavv"=>"MDAwMDAwMDAwMDAxMTA2Njk5NFg=", "protocol_version"=>"2", "directory_server_id"=>"f47ccba0-a98a-0138-0f10-00505690f91e"}");


    // Sca Params
    request.setScaExemption("trusted_merchant");
    request.setScaVisaMerchantId("00000000");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.init_recurring_sale3d(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "4012000000060085",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "business_attributes": {
      "event_start_date": "16-08-2020",
      "event_end_date": "25-08-2020",
      "event_organizer_id": "20192375",
      "event_id": "1912"
    },
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    },
    "mpi_params": {
      "eci": "05",
      "cavv": "MDAwMDAwMDAwMDAxMTA2Njk5NFg=",
      "protocol_version": "2",
      "directory_server_id": "f47ccba0-a98a-0138-0f10-00505690f91e"
    },
    "sca_params": {
      "exemption": "trusted_merchant",
      "visa_merchant_id": "00000000"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>init_recurring_sale3d</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>100</amount>
    <currency>USD</currency>
    <card_holder>Travis Pastrana</card_holder>
    <card_number>4012000000060085</card_number>
    <expiration_month>12</expiration_month>
    <expiration_year>2021</expiration_year>
    <cvv>834</cvv>
    <customer_email>travis@example.com</customer_email>
    <customer_phone>+1987987987987</customer_phone>
    <business_attributes>
      <event_start_date>16-08-2020</event_start_date>
      <event_end_date>25-08-2020</event_end_date>
      <event_organizer_id>20192375</event_organizer_id>
      <event_id>1912</event_id>
    </business_attributes>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>10178</zip_code>
      <city>Los Angeles</city>
      <state>CA</state>
      <country>US</country>
    </billing_address>
    <mpi_params>
      <eci>05</eci>
      <cavv>MDAwMDAwMDAwMDAxMTA2Njk5NFg=</cavv>
      <protocol_version>2</protocol_version>
      <directory_server_id>f47ccba0-a98a-0138-0f10-00505690f91e</directory_server_id>
    </mpi_params>
    <sca_params>
      <exemption>trusted_merchant</exemption>
      <visa_merchant_id>00000000</visa_merchant_id>
    </sca_params>
  </payment_transaction>'
```

Master Synchronous 3 D Sv2 Fully Authenticated Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Recurring\InitRecurringSale3D');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('5555555555559997')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')


        // MPI Synchronous 3DSv2
        ->setMpiEci('02')
        ->setMpiCavv('kQOImea0C/2ta1NZl4Hhwslmomqj')
        ->setMpiProtocolVersion('2')
        ->setMpiDirectoryServerId('f480e9b0-a98a-0138-0f10-00505690f91e');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.recurring.InitRecurringSale3D;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    InitRecurringSale3D request = new InitRecurringSale3D();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("5555555555559997");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");


    // Mpi Synchronous V2
    request.setMpiParams("{"eci"=>"02", "cavv"=>"kQOImea0C/2ta1NZl4Hhwslmomqj", "protocol_version"=>"2", "directory_server_id"=>"f480e9b0-a98a-0138-0f10-00505690f91e"}");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.init_recurring_sale3d(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "5555555555559997",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "business_attributes": {
      "event_start_date": "16-08-2020",
      "event_end_date": "25-08-2020",
      "event_organizer_id": "20192375",
      "event_id": "1912"
    },
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    },
    "mpi_params": {
      "eci": "02",
      "cavv": "kQOImea0C/2ta1NZl4Hhwslmomqj",
      "protocol_version": "2",
      "directory_server_id": "f480e9b0-a98a-0138-0f10-00505690f91e"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```curl
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>init_recurring_sale3d</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>100</amount>
    <currency>USD</currency>
    <card_holder>Travis Pastrana</card_holder>
    <card_number>5555555555559997</card_number>
    <expiration_month>12</expiration_month>
    <expiration_year>2021</expiration_year>
    <cvv>834</cvv>
    <customer_email>travis@example.com</customer_email>
    <customer_phone>+1987987987987</customer_phone>
    <business_attributes>
      <event_start_date>16-08-2020</event_start_date>
      <event_end_date>25-08-2020</event_end_date>
      <event_organizer_id>20192375</event_organizer_id>
      <event_id>1912</event_id>
    </business_attributes>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>10178</zip_code>
      <city>Los Angeles</city>
      <state>CA</state>
      <country>US</country>
    </billing_address>
    <mpi_params>
      <eci>02</eci>
      <cavv>kQOImea0C/2ta1NZl4Hhwslmomqj</cavv>
      <protocol_version>2</protocol_version>
      <directory_server_id>f480e9b0-a98a-0138-0f10-00505690f91e</directory_server_id>
    </mpi_params>
  </payment_transaction>'
```

Visa Synchronous 3 D Sv2 Attempted Authentication Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Recurring\InitRecurringSale3D');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('4012000000060085')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')


        // MPI Synchronous 3DSv2
        ->setMpiEci('06')
        ->setMpiCavv('MDAwMDAwMDAwMDAxMTA2Njk5NFg=')
        ->setMpiProtocolVersion('2')
        ->setMpiDirectoryServerId('f484f340-a98a-0138-0f10-00505690f91e')


        // Sca Params
        ->setScaExemption('trusted_merchant')
        ->setScaVisaMerchantId('00000000');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.recurring.InitRecurringSale3D;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    InitRecurringSale3D request = new InitRecurringSale3D();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("4012000000060085");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");


    // Mpi Synchronous V2
    request.setMpiParams("{"eci"=>"06", "cavv"=>"MDAwMDAwMDAwMDAxMTA2Njk5NFg=", "protocol_version"=>"2", "directory_server_id"=>"f484f340-a98a-0138-0f10-00505690f91e"}");


    // Sca Params
    request.setScaExemption("trusted_merchant");
    request.setScaVisaMerchantId("00000000");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.init_recurring_sale3d(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "4012000000060085",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "business_attributes": {
      "event_start_date": "16-08-2020",
      "event_end_date": "25-08-2020",
      "event_organizer_id": "20192375",
      "event_id": "1912"
    },
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    },
    "mpi_params": {
      "eci": "06",
      "cavv": "MDAwMDAwMDAwMDAxMTA2Njk5NFg=",
      "protocol_version": "2",
      "directory_server_id": "f484f340-a98a-0138-0f10-00505690f91e"
    },
    "sca_params": {
      "exemption": "trusted_merchant",
      "visa_merchant_id": "00000000"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>init_recurring_sale3d</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>100</amount>
    <currency>USD</currency>
    <card_holder>Travis Pastrana</card_holder>
    <card_number>4012000000060085</card_number>
    <expiration_month>12</expiration_month>
    <expiration_year>2021</expiration_year>
    <cvv>834</cvv>
    <customer_email>travis@example.com</customer_email>
    <customer_phone>+1987987987987</customer_phone>
    <business_attributes>
      <event_start_date>16-08-2020</event_start_date>
      <event_end_date>25-08-2020</event_end_date>
      <event_organizer_id>20192375</event_organizer_id>
      <event_id>1912</event_id>
    </business_attributes>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>10178</zip_code>
      <city>Los Angeles</city>
      <state>CA</state>
      <country>US</country>
    </billing_address>
    <mpi_params>
      <eci>06</eci>
      <cavv>MDAwMDAwMDAwMDAxMTA2Njk5NFg=</cavv>
      <protocol_version>2</protocol_version>
      <directory_server_id>f484f340-a98a-0138-0f10-00505690f91e</directory_server_id>
    </mpi_params>
    <sca_params>
      <exemption>trusted_merchant</exemption>
      <visa_merchant_id>00000000</visa_merchant_id>
    </sca_params>
  </payment_transaction>'
```

Master Synchronous 3 D Sv2 Attempted Authentication Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Recurring\InitRecurringSale3D');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('5555555555559997')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')


        // MPI Synchronous 3DSv2
        ->setMpiEci('01')
        ->setMpiCavv('kEOImea0C/2ta1NZl4Hhwslmomqj')
        ->setMpiProtocolVersion('2')
        ->setMpiDirectoryServerId('f4887c20-a98a-0138-0f10-00505690f91e');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.recurring.InitRecurringSale3D;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    InitRecurringSale3D request = new InitRecurringSale3D();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("5555555555559997");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");


    // Mpi Synchronous V2
    request.setMpiParams("{"eci"=>"01", "cavv"=>"kEOImea0C/2ta1NZl4Hhwslmomqj", "protocol_version"=>"2", "directory_server_id"=>"f4887c20-a98a-0138-0f10-00505690f91e"}");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.init_recurring_sale3d(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "5555555555559997",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "business_attributes": {
      "event_start_date": "16-08-2020",
      "event_end_date": "25-08-2020",
      "event_organizer_id": "20192375",
      "event_id": "1912"
    },
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    },
    "mpi_params": {
      "eci": "01",
      "cavv": "kEOImea0C/2ta1NZl4Hhwslmomqj",
      "protocol_version": "2",
      "directory_server_id": "f4887c20-a98a-0138-0f10-00505690f91e"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```curl
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>init_recurring_sale3d</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>100</amount>
    <currency>USD</currency>
    <card_holder>Travis Pastrana</card_holder>
    <card_number>5555555555559997</card_number>
    <expiration_month>12</expiration_month>
    <expiration_year>2021</expiration_year>
    <cvv>834</cvv>
    <customer_email>travis@example.com</customer_email>
    <customer_phone>+1987987987987</customer_phone>
    <business_attributes>
      <event_start_date>16-08-2020</event_start_date>
      <event_end_date>25-08-2020</event_end_date>
      <event_organizer_id>20192375</event_organizer_id>
      <event_id>1912</event_id>
    </business_attributes>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>10178</zip_code>
      <city>Los Angeles</city>
      <state>CA</state>
      <country>US</country>
    </billing_address>
    <mpi_params>
      <eci>01</eci>
      <cavv>kEOImea0C/2ta1NZl4Hhwslmomqj</cavv>
      <protocol_version>2</protocol_version>
      <directory_server_id>f4887c20-a98a-0138-0f10-00505690f91e</directory_server_id>
    </mpi_params>
  </payment_transaction>'
```

Asynchronous 3 D Sv2 Frictionless Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Recurring\InitRecurringSale3D');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('4012000000060085')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')
        ->setNotificationUrl('https://www.example.com/notification')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')


        // Threeds V2 Params
        ->setThreedsControl('{"device_type"=>"browser", "challenge_window_size"=>"full_screen", "challenge_indicator"=>"preference"}')
        ->setThreedsPurchase('{"category"=>"service"}')
        ->setThreedsRecurring('{"expiration_date"=>"16-01-2021", "frequency"=>30}')
        ->setThreedsMerchantRisk('{"shipping_indicator"=>"verified_address", "delivery_timeframe"=>"electronic", "reorder_items_indicator"=>"reordered", "pre_order_purchase_indicator"=>"merchandise_available",
        ->setThreedsCardHolderAccount('{"creation_date"=>"16-07-2019", "update_indicator"=>"more_than_60days", "last_change_date"=>"16-04-2020", "password_change_indicator"=>"no_change", "password_change_date"
        ->setThreedsBrowser('{"accept_header"=>"Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser", "java_enabled"=>"true", "language"=>"en-GB", "color_depth"=>"
        ->setThreedsSdk('{"interface"=>"native", "ui_types"=>{"ui_type"=>"multi_select"}, "application_id"=>"fc1650c0-5778-0138-8205-2cbc32a32d65", "encrypted_data"=>"encrypted-data-here", "ephemeral_public_ke

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.recurring.InitRecurringSale3D;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;
import java.net.MalformedURLException;
import java.net.URL;

public class GenesisExample {
  public static void main() throws MalformedURLException {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    InitRecurringSale3D request = new InitRecurringSale3D();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("4012000000060085");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");
    request.setNotificationUrl(new URL("https://www.example.com/notification"));
    request.setReturnSuccessUrl(new URL("http://www.example.com/success"));
    request.setReturnFailureUrl(new URL("http://www.example.com/failure"));


    // Threeds V2 Params
    request.setThreedsControl("{"device_type"=>"browser", "challenge_window_size"=>"full_screen", "challenge_indicator"=>"preference"}");
    request.setThreedsPurchase("{"category"=>"service"}");
    request.setThreedsRecurring("{"expiration_date"=>"16-01-2021", "frequency"=>30}");
    request.setThreedsMerchantRisk("{"shipping_indicator"=>"verified_address", "delivery_timeframe"=>"electronic", "reorder_items_indicator"=>"reordered", "pre_order_purchase_indicator"=>"merchandise_available
    request.setThreedsCardHolderAccount("{"creation_date"=>"16-07-2019", "update_indicator"=>"more_than_60days", "last_change_date"=>"16-04-2020", "password_change_indicator"=>"no_change", "password_change_da
    request.setThreedsBrowser("{"accept_header"=>"Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser", "java_enabled"=>"true", "language"=>"en-GB", "color_depth"
    request.setThreedsSdk("{"interface"=>"native", "ui_types"=>{"ui_type"=>"multi_select"}, "application_id"=>"fc1650c0-5778-0138-8205-2cbc32a32d65", "encrypted_data"=>"encrypted-data-here", "ephemeral_public_

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```python
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.init_recurring_sale3d(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "4012000000060085",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "billing_address": {
```

```
      "billing_address": {
        "first_name": "Travis",
        "last_name": "Pastrana",
        "address1": "Muster Str. 12",
        "zip_code": "10178",
        "city": "Los Angeles",
        "state": "CA",
        "country": "US"
      },
      "notification_url": "https://www.example.com/notification",
      "return_success_url": "http://www.example.com/success",
      "return_failure_url": "http://www.example.com/failure",
      "threeds_v2_params": {
        "control": {
          "device_type": "browser",
          "challenge_window_size": "full_screen",
          "challenge_indicator": "preference"
        },
        "purchase": {
          "category": "service"
        },
        "recurring": {
          "expiration_date": "16-01-2021",
          "frequency": 30
        },
        "merchant_risk": {
          "shipping_indicator": "verified_address",
          "delivery_timeframe": "electronic",
          "reorder_items_indicator": "reordered",
          "pre_order_purchase_indicator": "merchandise_available",
          "pre_order_date": "16-08-2020",
          "gift_card": "true",
          "gift_card_count": "2"
        },
        "card_holder_account": {
          "creation_date": "16-07-2019",
          "update_indicator": "more_than_60days",
          "last_change_date": "16-04-2020",
          "password_change_indicator": "no_change",
          "password_change_date": "01-07-2020",
          "shipping_address_usage_indicator": "current_transaction",
          "shipping_address_date_first_used": "11-07-2020",
          "transactions_activity_last_24_hours": "2",
          "transactions_activity_previous_year": "10",
          "provision_attempts_last_24_hours": "1",
          "purchases_count_last_6_months": "5",
          "suspicious_activity_indicator": "no_suspicious_observed",
          "registration_indicator": "30_to_60_days",
          "registration_date": "16-07-2018"
        },
        "browser": {
          "accept_header": "Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser",
          "java_enabled": "true",
          "language": "en-GB",
          "color_depth": "32",
          "screen_height": "900",
          "screen_width": "1440",
          "time_zone_offset": "-120",
          "user_agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.97 Safari/537.36"
        },
        "sdk": {
          "interface": "native",
          "ui_types": {
            "ui_type": "multi_select"
          },
          "application_id": "fc1650c0-5778-0138-8205-2cbc32a32d65",
          "encrypted_data": "encrypted-data-here",
          "ephemeral_public_key_pair": "public-key-pair",
          "max_timeout": "10",
          "reference_number": "sdk-reference-number-here"
        }
      }
    }
  ).send()
    .then(success)
    .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>init_recurring_sale3d</transaction_type>
   <transaction_id>43671</transaction_id>
   <usage>40208 concert tickets</usage>
   <remote_ip>245.253.2.12</remote_ip>
   <amount>100</amount>
   <currency>USD</currency>
   <card_holder>Travis Pastrana</card_holder>
   <card_number>4012000000060085</card_number>
   <expiration_month>12</expiration_month>
   <expiration_year>2021</expiration_year>
   <cvv>834</cvv>
   <customer_email>travis@example.com</customer_email>
   <customer_phone>+1987987987987</customer_phone>
   <billing_address>
     <first_name>Travis</first_name>
     <last_name>Pastrana</last_name>
     <address1>Muster Str. 12</address1>
     <zip_code>10178</zip_code>
     <city>Los Angeles</city>
     <state>CA</state>
     <country>US</country>
   </billing_address>
   <notification_url>https://www.example.com/notification</notification_url>
   <return_success_url>http://www.example.com/success</return_success_url>
   <return_failure_url>http://www.example.com/failure</return_failure_url>
   <threeds_v2_params>
     <control>
       <device_type>browser</device_type>
       <challenge_window_size>full_screen</challenge_window_size>
       <challenge_indicator>preference</challenge_indicator>
     </control>
     <purchase>
       <category>service</category>
     </purchase>
     <recurring>
       <expiration_date>16-01-2021</expiration_date>
       <frequency>30</frequency>
     </recurring>
     <merchant_risk>
       <shipping_indicator>verified_address</shipping_indicator>
       <delivery_timeframe>electronic</delivery_timeframe>
       <reorder_items_indicator>reordered</reorder_items_indicator>
       <pre_order_purchase_indicator>merchandise_available</pre_order_purchase_indicator>
       <pre_order_date>16-08-2020</pre_order_date>
       <gift_card>true</gift_card>
       <gift_card_count>2</gift_card_count>
     </merchant_risk>
     <card_holder_account>
       <creation_date>16-07-2019</creation_date>
       <update_indicator>more_than_60days</update_indicator>
       <last_change_date>16-04-2020</last_change_date>
       <password_change_indicator>no_change</password_change_indicator>
       <password_change_date>01-07-2020</password_change_date>
       <shipping_address_usage_indicator>current_transaction</shipping_address_usage_indicator>
       <shipping_address_date_first_used>11-07-2020</shipping_address_date_first_used>
       <transactions_activity_last_24_hours>2</transactions_activity_last_24_hours>
       <transactions_activity_previous_year>10</transactions_activity_previous_year>
       <provision_attempts_last_24_hours>1</provision_attempts_last_24_hours>
       <purchases_count_last_6_months>5</purchases_count_last_6_months>
       <suspicious_activity_indicator>no_suspicious_observed</suspicious_activity_indicator>
       <registration_indicator>30_to_60_days</registration_indicator>
       <registration_date>16-07-2018</registration_date>
     </card_holder_account>
     <browser>
       <accept_header>Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser</accept_header>
       <java_enabled>true</java_enabled>
       <language>en-GB</language>
       <color_depth>32</color_depth>
       <screen_height>900</screen_height>
       <screen_width>1440</screen_width>
       <time_zone_offset>-120</time_zone_offset>
       <user_agent>Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.97 Safari/537.36</user_agent>
     </browser>
     <sdk>
       <interface>native</interface>
       <ui_types>
         <ui_type>multi_select</ui_type>
       </ui_types>
       <application_id>fc1650c0-5778-0138-8205-2cbc32a32d65</application_id>
       <encrypted_data>encrypted-data-here</encrypted_data>
       <ephemeral_public_key_pair>public-key-pair</ephemeral_public_key_pair>
       <max_timeout>10</max_timeout>
       <reference_number>sdk-reference-number-here</reference_number>
     </sdk>
   </threeds_v2_params>
 </payment_transaction>'
```

Asynchronous 3 D Sv2 Frictionless With 3ds Method Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Recurring\InitRecurringSale3D');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('4066330000000004')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')
        ->setNotificationUrl('https://www.example.com/notification')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')


        // Threeds V2 Params
        ->setThreedsControl('{"device_type"=>"browser", "challenge_window_size"=>"full_screen", "challenge_indicator"=>"preference"}')
        ->setThreedsPurchase('{"category"=>"service"}')
        ->setThreedsRecurring('{"expiration_date"=>"16-01-2021", "frequency"=>30}')
        ->setThreedsMerchantRisk('{"shipping_indicator"=>"verified_address", "delivery_timeframe"=>"electronic", "reorder_items_indicator"=>"reordered", "pre_order_purchase_indicator"=>"merchandise_available",
        ->setThreedsCardHolderAccount('{"creation_date"=>"16-07-2019", "update_indicator"=>"more_than_60days", "last_change_date"=>"16-04-2020", "password_change_indicator"=>"no_change", "password_change_date"
        ->setThreedsBrowser('{"accept_header"=>"Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser", "java_enabled"=>"true", "language"=>"en-GB", "color_depth"=>"
        ->setThreedsSdk('{"interface"=>"native", "ui_types"=>{"ui_type"=>"multi_select"}, "application_id"=>"fc1650c0-5778-0138-8205-2cbc32a32d65", "encrypted_data"=>"encrypted-data-here", "ephemeral_public_ke

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.recurring.InitRecurringSale3D;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;
import java.net.MalformedURLException;
import java.net.URL;

public class GenesisExample {
  public static void main() throws MalformedURLException {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    InitRecurringSale3D request = new InitRecurringSale3D();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("4066330000000004");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");
    request.setNotificationUrl(new URL("https://www.example.com/notification"));
    request.setReturnSuccessUrl(new URL("http://www.example.com/success"));
    request.setReturnFailureUrl(new URL("http://www.example.com/failure"));


    // Threeds V2 Params
    request.setThreedsControl("{"device_type"=>"browser", "challenge_window_size"=>"full_screen", "challenge_indicator"=>"preference"}");
    request.setThreedsPurchase("{"category"=>"service"}");
    request.setThreedsRecurring("{"expiration_date"=>"16-01-2021", "frequency"=>30}");
    request.setThreedsMerchantRisk("{"shipping_indicator"=>"verified_address", "delivery_timeframe"=>"electronic", "reorder_items_indicator"=>"reordered", "pre_order_purchase_indicator"=>"merchandise_available
    request.setThreedsCardHolderAccount("{"creation_date"=>"16-07-2019", "update_indicator"=>"more_than_60days", "last_change_date"=>"16-04-2020", "password_change_indicator"=>"no_change", "password_change_da
    request.setThreedsBrowser("{"accept_header"=>"Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser", "java_enabled"=>"true", "language"=>"en-GB", "color_depth"
    request.setThreedsSdk("{"interface"=>"native", "ui_types"=>{"ui_type"=>"multi_select"}, "application_id"=>"fc1650c0-5778-0138-8205-2cbc32a32d65", "encrypted_data"=>"encrypted-data-here", "ephemeral_public_

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.init_recurring_sale3d(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "4066330000000004",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "billing_address": {
```

```
      "billing_address": {
        "first_name": "Travis",
        "last_name": "Pastrana",
        "address1": "Muster Str. 12",
        "zip_code": "10178",
        "city": "Los Angeles",
        "state": "CA",
        "country": "US"
      },
      "notification_url": "https://www.example.com/notification",
      "return_success_url": "http://www.example.com/success",
      "return_failure_url": "http://www.example.com/failure",
      "threeds_v2_params": {
        "control": {
          "device_type": "browser",
          "challenge_window_size": "full_screen",
          "challenge_indicator": "preference"
        },
        "purchase": {
          "category": "service"
        },
        "recurring": {
          "expiration_date": "16-01-2021",
          "frequency": 30
        },
        "merchant_risk": {
          "shipping_indicator": "verified_address",
          "delivery_timeframe": "electronic",
          "reorder_items_indicator": "reordered",
          "pre_order_purchase_indicator": "merchandise_available",
          "pre_order_date": "16-08-2020",
          "gift_card": "true",
          "gift_card_count": "2"
        },
        "card_holder_account": {
          "creation_date": "16-07-2019",
          "update_indicator": "more_than_60days",
          "last_change_date": "16-04-2020",
          "password_change_indicator": "no_change",
          "password_change_date": "01-07-2020",
          "shipping_address_usage_indicator": "current_transaction",
          "shipping_address_date_first_used": "11-07-2020",
          "transactions_activity_last_24_hours": "2",
          "transactions_activity_previous_year": "10",
          "provision_attempts_last_24_hours": "1",
          "purchases_count_last_6_months": "5",
          "suspicious_activity_indicator": "no_suspicious_observed",
          "registration_indicator": "30_to_60_days",
          "registration_date": "16-07-2018"
        },
        "browser": {
          "accept_header": "Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser",
          "java_enabled": "true",
          "language": "en-GB",
          "color_depth": "32",
          "screen_height": "900",
          "screen_width": "1440",
          "time_zone_offset": "-120",
          "user_agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.97 Safari/537.36"
        },
        "sdk": {
          "interface": "native",
          "ui_types": {
            "ui_type": "multi_select"
          },
          "application_id": "fc1650c0-5778-0138-8205-2cbc32a32d65",
          "encrypted_data": "encrypted-data-here",
          "ephemeral_public_key_pair": "public-key-pair",
          "max_timeout": "10",
          "reference_number": "sdk-reference-number-here"
        }
      }
    }
  ).send()
    .then(success)
    .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>init_recurring_sale3d</transaction_type>
   <transaction_id>43671</transaction_id>
   <usage>40208 concert tickets</usage>
   <remote_ip>245.253.2.12</remote_ip>
   <amount>100</amount>
   <currency>USD</currency>
   <card_holder>Travis Pastrana</card_holder>
   <card_number>4066330000000004</card_number>
   <expiration_month>12</expiration_month>
   <expiration_year>2021</expiration_year>
   <cvv>834</cvv>
   <customer_email>travis@example.com</customer_email>
   <customer_phone>+1987987987987</customer_phone>
   <billing_address>
     <first_name>Travis</first_name>
     <last_name>Pastrana</last_name>
     <address1>Muster Str. 12</address1>
     <zip_code>10178</zip_code>
     <city>Los Angeles</city>
     <state>CA</state>
     <country>US</country>
   </billing_address>
   <notification_url>https://www.example.com/notification</notification_url>
   <return_success_url>http://www.example.com/success</return_success_url>
   <return_failure_url>http://www.example.com/failure</return_failure_url>
   <threeds_v2_params>
     <control>
       <device_type>browser</device_type>
       <challenge_window_size>full_screen</challenge_window_size>
       <challenge_indicator>preference</challenge_indicator>
     </control>
     <purchase>
       <category>service</category>
     </purchase>
     <recurring>
       <expiration_date>16-01-2021</expiration_date>
       <frequency>30</frequency>
     </recurring>
     <merchant_risk>
       <shipping_indicator>verified_address</shipping_indicator>
       <delivery_timeframe>electronic</delivery_timeframe>
       <reorder_items_indicator>reordered</reorder_items_indicator>
       <pre_order_purchase_indicator>merchandise_available</pre_order_purchase_indicator>
       <pre_order_date>16-08-2020</pre_order_date>
       <gift_card>true</gift_card>
       <gift_card_count>2</gift_card_count>
     </merchant_risk>
     <card_holder_account>
       <creation_date>16-07-2019</creation_date>
       <update_indicator>more_than_60days</update_indicator>
       <last_change_date>16-04-2020</last_change_date>
       <password_change_indicator>no_change</password_change_indicator>
       <password_change_date>01-07-2020</password_change_date>
       <shipping_address_usage_indicator>current_transaction</shipping_address_usage_indicator>
       <shipping_address_date_first_used>11-07-2020</shipping_address_date_first_used>
       <transactions_activity_last_24_hours>2</transactions_activity_last_24_hours>
       <transactions_activity_previous_year>10</transactions_activity_previous_year>
       <provision_attempts_last_24_hours>1</provision_attempts_last_24_hours>
       <purchases_count_last_6_months>5</purchases_count_last_6_months>
       <suspicious_activity_indicator>no_suspicious_observed</suspicious_activity_indicator>
       <registration_indicator>30_to_60_days</registration_indicator>
       <registration_date>16-07-2018</registration_date>
     </card_holder_account>
     <browser>
       <accept_header>Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser</accept_header>
       <java_enabled>true</java_enabled>
       <language>en-GB</language>
       <color_depth>32</color_depth>
       <screen_height>900</screen_height>
       <screen_width>1440</screen_width>
       <time_zone_offset>-120</time_zone_offset>
       <user_agent>Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.97 Safari/537.36</user_agent>
     </browser>
     <sdk>
       <interface>native</interface>
       <ui_types>
         <ui_type>multi_select</ui_type>
       </ui_types>
       <application_id>fc1650c0-5778-0138-8205-2cbc32a32d65</application_id>
       <encrypted_data>encrypted-data-here</encrypted_data>
       <ephemeral_public_key_pair>public-key-pair</ephemeral_public_key_pair>
       <max_timeout>10</max_timeout>
       <reference_number>sdk-reference-number-here</reference_number>
     </sdk>
   </threeds_v2_params>
 </payment_transaction>'
```

Asynchronous 3 D Sv2 Challenge Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Recurring\InitRecurringSale3D');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('4918190000000002')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')
        ->setNotificationUrl('https://www.example.com/notification')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')


        // Threeds V2 Params
        ->setThreedsControl('{"device_type"="browser", "challenge_window_size"="full_screen", "challenge_indicator"="preference"}')
        ->setThreedsPurchase('{"category"="service"}')
        ->setThreedsRecurring('{"expiration_date"="16-01-2021", "frequency"=>30}')
        ->setThreedsMerchantRisk('{"shipping_indicator"="verified_address", "delivery_timeframe"="electronic", "reorder_items_indicator"="reordered", "pre_order_purchase_indicator"="merchandise_available",
        ->setThreedsCardHolderAccount('{"creation_date"="16-07-2019", "update_indicator"="more_than_60days", "last_change_date"="16-04-2020", "password_change_indicator"="no_change", "password_change_date"
        ->setThreedsBrowser('{"accept_header"="Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser", "java_enabled"="true", "language"="en-GB", "color_depth"="
        ->setThreedsSdk('{"interface"="native", "ui_types"={"ui_type"="multi_select"}, "application_id"="fc1650c0-5778-0138-8205-2cbc32a32d65", "encrypted_data"="encrypted-data-here", "ephemeral_public_ke

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.recurring.InitRecurringSale3D;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;
import java.net.MalformedURLException;
import java.net.URL;

public class GenesisExample {
  public static void main() throws MalformedURLException {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    InitRecurringSale3D request = new InitRecurringSale3D();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("4918190000000002");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");
    request.setNotificationUrl(new URL("https://www.example.com/notification"));
    request.setReturnSuccessUrl(new URL("http://www.example.com/success"));
    request.setReturnFailureUrl(new URL("http://www.example.com/failure"));


    // Threeds V2 Params
    request.setThreedsControl("{"device_type"=>"browser", "challenge_window_size"=>"full_screen", "challenge_indicator"=>"preference"}");
    request.setThreedsPurchase("{"category"=>"service"}");
    request.setThreedsRecurring("{"expiration_date"=>"16-01-2021", "frequency"=>30}");
    request.setThreedsMerchantRisk("{"shipping_indicator"=>"verified_address", "delivery_timeframe"=>"electronic", "reorder_items_indicator"=>"reordered", "pre_order_purchase_indicator"=>"merchandise_available
    request.setThreedsCardHolderAccount("{"creation_date"=>"16-07-2019", "update_indicator"=>"more_than_60days", "last_change_date"=>"16-04-2020", "password_change_indicator"=>"no_change", "password_change_da
    request.setThreedsBrowser("{"accept_header"=>"Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser", "java_enabled"=>"true", "language"=>"en-GB", "color_depth"
    request.setThreedsSdk("{"interface"=>"native", "ui_types"=>{"ui_type"=>"multi_select"}, "application_id"=>"fc1650c0-5778-0138-8205-2cbc32a32d65", "encrypted_data"=>"encrypted-data-here", "ephemeral_public_

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```python
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.init_recurring_sale3d(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "4918190000000002",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "billing_address": {
```

```
      "billing_address": {
        "first_name": "Travis",
        "last_name": "Pastrana",
        "address1": "Muster Str. 12",
        "zip_code": "10178",
        "city": "Los Angeles",
        "state": "CA",
        "country": "US"
      },
      "notification_url": "https://www.example.com/notification",
      "return_success_url": "http://www.example.com/success",
      "return_failure_url": "http://www.example.com/failure",
      "threeds_v2_params": {
        "control": {
          "device_type": "browser",
          "challenge_window_size": "full_screen",
          "challenge_indicator": "preference"
        },
        "purchase": {
          "category": "service"
        },
        "recurring": {
          "expiration_date": "16-01-2021",
          "frequency": 30
        },
        "merchant_risk": {
          "shipping_indicator": "verified_address",
          "delivery_timeframe": "electronic",
          "reorder_items_indicator": "reordered",
          "pre_order_purchase_indicator": "merchandise_available",
          "pre_order_date": "16-08-2020",
          "gift_card": "true",
          "gift_card_count": "2"
        },
        "card_holder_account": {
          "creation_date": "16-07-2019",
          "update_indicator": "more_than_60days",
          "last_change_date": "16-04-2020",
          "password_change_indicator": "no_change",
          "password_change_date": "01-07-2020",
          "shipping_address_usage_indicator": "current_transaction",
          "shipping_address_date_first_used": "11-07-2020",
          "transactions_activity_last_24_hours": "2",
          "transactions_activity_previous_year": "10",
          "provision_attempts_last_24_hours": "1",
          "purchases_count_last_6_months": "5",
          "suspicious_activity_indicator": "no_suspicious_observed",
          "registration_indicator": "30_to_60_days",
          "registration_date": "16-07-2018"
        },
        "browser": {
          "accept_header": "Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser",
          "java_enabled": "true",
          "language": "en-GB",
          "color_depth": "32",
          "screen_height": "900",
          "screen_width": "1440",
          "time_zone_offset": "-120",
          "user_agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.97 Safari/537.36"
        },
        "sdk": {
          "interface": "native",
          "ui_types": {
            "ui_type": "multi_select"
          },
          "application_id": "fc1650c0-5778-0138-8205-2cbc32a32d65",
          "encrypted_data": "encrypted-data-here",
          "ephemeral_public_key_pair": "public-key-pair",
          "max_timeout": "10",
          "reference_number": "sdk-reference-number-here"
        }
      }
    }
  ).send()
    .then(success)
    .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>init_recurring_sale3d</transaction_type>
   <transaction_id>43671</transaction_id>
   <usage>40208 concert tickets</usage>
   <remote_ip>245.253.2.12</remote_ip>
   <amount>100</amount>
   <currency>USD</currency>
   <card_holder>Travis Pastrana</card_holder>
   <card_number>4918190000000002</card_number>
   <expiration_month>12</expiration_month>
   <expiration_year>2021</expiration_year>
   <cvv>834</cvv>
   <customer_email>travis@example.com</customer_email>
   <customer_phone>+1987987987987</customer_phone>
   <billing_address>
     <first_name>Travis</first_name>
     <last_name>Pastrana</last_name>
     <address1>Muster Str. 12</address1>
     <zip_code>10178</zip_code>
     <city>Los Angeles</city>
     <state>CA</state>
     <country>US</country>
   </billing_address>
   <notification_url>https://www.example.com/notification</notification_url>
   <return_success_url>http://www.example.com/success</return_success_url>
   <return_failure_url>http://www.example.com/failure</return_failure_url>
   <threeds_v2_params>
     <control>
       <device_type>browser</device_type>
       <challenge_window_size>full_screen</challenge_window_size>
       <challenge_indicator>preference</challenge_indicator>
     </control>
     <purchase>
       <category>service</category>
     </purchase>
     <recurring>
       <expiration_date>16-01-2021</expiration_date>
       <frequency>30</frequency>
     </recurring>
     <merchant_risk>
       <shipping_indicator>verified_address</shipping_indicator>
       <delivery_timeframe>electronic</delivery_timeframe>
       <reorder_items_indicator>reordered</reorder_items_indicator>
       <pre_order_purchase_indicator>merchandise_available</pre_order_purchase_indicator>
       <pre_order_date>16-08-2020</pre_order_date>
       <gift_card>true</gift_card>
       <gift_card_count>2</gift_card_count>
     </merchant_risk>
     <card_holder_account>
       <creation_date>16-07-2019</creation_date>
       <update_indicator>more_than_60days</update_indicator>
       <last_change_date>16-04-2020</last_change_date>
       <password_change_indicator>no_change</password_change_indicator>
       <password_change_date>01-07-2020</password_change_date>
       <shipping_address_usage_indicator>current_transaction</shipping_address_usage_indicator>
       <shipping_address_date_first_used>11-07-2020</shipping_address_date_first_used>
       <transactions_activity_last_24_hours>2</transactions_activity_last_24_hours>
       <transactions_activity_previous_year>10</transactions_activity_previous_year>
       <provision_attempts_last_24_hours>1</provision_attempts_last_24_hours>
       <purchases_count_last_6_months>5</purchases_count_last_6_months>
       <suspicious_activity_indicator>no_suspicious_observed</suspicious_activity_indicator>
       <registration_indicator>30_to_60_days</registration_indicator>
       <registration_date>16-07-2018</registration_date>
     </card_holder_account>
     <browser>
       <accept_header>Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser</accept_header>
       <java_enabled>true</java_enabled>
       <language>en-GB</language>
       <color_depth>32</color_depth>
       <screen_height>900</screen_height>
       <screen_width>1440</screen_width>
       <time_zone_offset>-120</time_zone_offset>
       <user_agent>Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.97 Safari/537.36</user_agent>
     </browser>
     <sdk>
       <interface>native</interface>
       <ui_types>
         <ui_type>multi_select</ui_type>
       </ui_types>
       <application_id>fc1650c0-5778-0138-8205-2cbc32a32d65</application_id>
       <encrypted_data>encrypted-data-here</encrypted_data>
       <ephemeral_public_key_pair>public-key-pair</ephemeral_public_key_pair>
       <max_timeout>10</max_timeout>
       <reference_number>sdk-reference-number-here</reference_number>
     </sdk>
   </threeds_v2_params>
 </payment_transaction>'
```

Asynchronous 3 D Sv2 Challenge With 3ds Method Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Recurring\InitRecurringSale3D');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('4938730000000001')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')
        ->setNotificationUrl('https://www.example.com/notification')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')


        // Threeds V2 Params
        ->setThreedsControl('{"device_type"=>"browser", "challenge_window_size"=>"full_screen", "challenge_indicator"=>"preference"}')
        ->setThreedsPurchase('{"category"=>"service"}')
        ->setThreedsRecurring('{"expiration_date"=>"16-01-2021", "frequency"=>30}')
        ->setThreedsMerchantRisk('{"shipping_indicator"=>"verified_address", "delivery_timeframe"=>"electronic", "reorder_items_indicator"=>"reordered", "pre_order_purchase_indicator"=>"merchandise_available",
        ->setThreedsCardHolderAccount('{"creation_date"=>"16-07-2019", "update_indicator"=>"more_than_60days", "last_change_date"=>"16-04-2020", "password_change_indicator"=>"no_change", "password_change_date"
        ->setThreedsBrowser('{"accept_header"=>"Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser", "java_enabled"=>"true", "language"=>"en-GB", "color_depth"=>"
        ->setThreedsSdk('{"interface"=>"native", "ui_types"=>{"ui_type"=>"multi_select"}, "application_id"=>"fc1650c0-5778-0138-8205-2cbc32a32d65", "encrypted_data"=>"encrypted-data-here", "ephemeral_public_ke

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.recurring.InitRecurringSale3D;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;
import java.net.MalformedURLException;
import java.net.URL;

public class GenesisExample {
  public static void main() throws MalformedURLException {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    InitRecurringSale3D request = new InitRecurringSale3D();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("4938730000000001");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");
    request.setNotificationUrl(new URL("https://www.example.com/notification"));
    request.setReturnSuccessUrl(new URL("http://www.example.com/success"));
    request.setReturnFailureUrl(new URL("http://www.example.com/failure"));


    // Threeds V2 Params
    request.setThreedsControl("{"device_type"=>"browser", "challenge_window_size"=>"full_screen", "challenge_indicator"=>"preference"}");
    request.setThreedsPurchase("{"category"=>"service"}");
    request.setThreedsRecurring("{"expiration_date"=>"16-01-2021", "frequency"=>30}");
    request.setThreedsMerchantRisk("{"shipping_indicator"=>"verified_address", "delivery_timeframe"=>"electronic", "reorder_items_indicator"=>"reordered", "pre_order_purchase_indicator"=>"merchandise_available
    request.setThreedsCardHolderAccount("{"creation_date"=>"16-07-2019", "update_indicator"=>"more_than_60days", "last_change_date"=>"16-04-2020", "password_change_indicator"=>"no_change", "password_change_da
    request.setThreedsBrowser("{"accept_header"=>"Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser", "java_enabled"=>"true", "language"=>"en-GB", "color_depth"
    request.setThreedsSdk("{"interface"=>"native", "ui_types"=>{"ui_type"=>"multi_select"}, "application_id"=>"fc1650c0-5778-0138-8205-2cbc32a32d65", "encrypted_data"=>"encrypted-data-here", "ephemeral_public_

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.init_recurring_sale3d(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "4938730000000001",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "billing_address": {
```

```
      "billing_address": {
        "first_name": "Travis",
        "last_name": "Pastrana",
        "address1": "Muster Str. 12",
        "zip_code": "10178",
        "city": "Los Angeles",
        "state": "CA",
        "country": "US"
      },
      "notification_url": "https://www.example.com/notification",
      "return_success_url": "http://www.example.com/success",
      "return_failure_url": "http://www.example.com/failure",
      "threeds_v2_params": {
        "control": {
          "device_type": "browser",
          "challenge_window_size": "full_screen",
          "challenge_indicator": "preference"
        },
        "purchase": {
          "category": "service"
        },
        "recurring": {
          "expiration_date": "16-01-2021",
          "frequency": 30
        },
        "merchant_risk": {
          "shipping_indicator": "verified_address",
          "delivery_timeframe": "electronic",
          "reorder_items_indicator": "reordered",
          "pre_order_purchase_indicator": "merchandise_available",
          "pre_order_date": "16-08-2020",
          "gift_card": "true",
          "gift_card_count": "2"
        },
        "card_holder_account": {
          "creation_date": "16-07-2019",
          "update_indicator": "more_than_60days",
          "last_change_date": "16-04-2020",
          "password_change_indicator": "no_change",
          "password_change_date": "01-07-2020",
          "shipping_address_usage_indicator": "current_transaction",
          "shipping_address_date_first_used": "11-07-2020",
          "transactions_activity_last_24_hours": "2",
          "transactions_activity_previous_year": "10",
          "provision_attempts_last_24_hours": "1",
          "purchases_count_last_6_months": "5",
          "suspicious_activity_indicator": "no_suspicious_observed",
          "registration_indicator": "30_to_60_days",
          "registration_date": "16-07-2018"
        },
        "browser": {
          "accept_header": "Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser",
          "java_enabled": "true",
          "language": "en-GB",
          "color_depth": "32",
          "screen_height": "900",
          "screen_width": "1440",
          "time_zone_offset": "-120",
          "user_agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.97 Safari/537.36"
        },
        "sdk": {
          "interface": "native",
          "ui_types": {
            "ui_type": "multi_select"
          },
          "application_id": "fc1650c0-5778-0138-8205-2cbc32a32d65",
          "encrypted_data": "encrypted-data-here",
          "ephemeral_public_key_pair": "public-key-pair",
          "max_timeout": "10",
          "reference_number": "sdk-reference-number-here"
        }
      }
    }
  ).send()
    .then(success)
    .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>init_recurring_sale3d</transaction_type>
   <transaction_id>43671</transaction_id>
   <usage>40208 concert tickets</usage>
   <remote_ip>245.253.2.12</remote_ip>
   <amount>100</amount>
   <currency>USD</currency>
   <card_holder>Travis Pastrana</card_holder>
   <card_number>4938730000000001</card_number>
   <expiration_month>12</expiration_month>
   <expiration_year>2021</expiration_year>
   <cvv>834</cvv>
   <customer_email>travis@example.com</customer_email>
   <customer_phone>+1987987987987</customer_phone>
   <billing_address>
     <first_name>Travis</first_name>
     <last_name>Pastrana</last_name>
     <address1>Muster Str. 12</address1>
     <zip_code>10178</zip_code>
     <city>Los Angeles</city>
     <state>CA</state>
     <country>US</country>
   </billing_address>
   <notification_url>https://www.example.com/notification</notification_url>
   <return_success_url>http://www.example.com/success</return_success_url>
   <return_failure_url>http://www.example.com/failure</return_failure_url>
   <threeds_v2_params>
     <control>
       <device_type>browser</device_type>
       <challenge_window_size>full_screen</challenge_window_size>
       <challenge_indicator>preference</challenge_indicator>
     </control>
     <purchase>
       <category>service</category>
     </purchase>
     <recurring>
       <expiration_date>16-01-2021</expiration_date>
       <frequency>30</frequency>
     </recurring>
     <merchant_risk>
       <shipping_indicator>verified_address</shipping_indicator>
       <delivery_timeframe>electronic</delivery_timeframe>
       <reorder_items_indicator>reordered</reorder_items_indicator>
       <pre_order_purchase_indicator>merchandise_available</pre_order_purchase_indicator>
       <pre_order_date>16-08-2020</pre_order_date>
       <gift_card>true</gift_card>
       <gift_card_count>2</gift_card_count>
     </merchant_risk>
     <card_holder_account>
       <creation_date>16-07-2019</creation_date>
       <update_indicator>more_than_60days</update_indicator>
       <last_change_date>16-04-2020</last_change_date>
       <password_change_indicator>no_change</password_change_indicator>
       <password_change_date>01-07-2020</password_change_date>
       <shipping_address_usage_indicator>current_transaction</shipping_address_usage_indicator>
       <shipping_address_date_first_used>11-07-2020</shipping_address_date_first_used>
       <transactions_activity_last_24_hours>2</transactions_activity_last_24_hours>
       <transactions_activity_previous_year>10</transactions_activity_previous_year>
       <provision_attempts_last_24_hours>1</provision_attempts_last_24_hours>
       <purchases_count_last_6_months>5</purchases_count_last_6_months>
       <suspicious_activity_indicator>no_suspicious_observed</suspicious_activity_indicator>
       <registration_indicator>30_to_60_days</registration_indicator>
       <registration_date>16-07-2018</registration_date>
     </card_holder_account>
     <browser>
       <accept_header>Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser</accept_header>
       <java_enabled>true</java_enabled>
       <language>en-GB</language>
       <color_depth>32</color_depth>
       <screen_height>900</screen_height>
       <screen_width>1440</screen_width>
       <time_zone_offset>-120</time_zone_offset>
       <user_agent>Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.97 Safari/537.36</user_agent>
     </browser>
     <sdk>
       <interface>native</interface>
       <ui_types>
         <ui_type>multi_select</ui_type>
       </ui_types>
       <application_id>fc1650c0-5778-0138-8205-2cbc32a32d65</application_id>
       <encrypted_data>encrypted-data-here</encrypted_data>
       <ephemeral_public_key_pair>public-key-pair</ephemeral_public_key_pair>
       <max_timeout>10</max_timeout>
       <reference_number>sdk-reference-number-here</reference_number>
     </sdk>
   </threeds_v2_params>
 </payment_transaction>'
```

Asynchronous 3 D Sv2 Fallback 3dsv1 Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Recurring\InitRecurringSale3D');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('4901170000000003')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')
        ->setNotificationUrl('https://www.example.com/notification')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')


        // Threeds V2 Params
        ->setThreedsControl('{"device_type"=>"browser", "challenge_window_size"=>"full_screen", "challenge_indicator"=>"preference"}')
        ->setThreedsPurchase('{"category"=>"service"}')
        ->setThreedsRecurring('{"expiration_date"=>"16-01-2021", "frequency"=>30}')
        ->setThreedsMerchantRisk('{"shipping_indicator"=>"verified_address", "delivery_timeframe"=>"electronic", "reorder_items_indicator"=>"reordered", "pre_order_purchase_indicator"=>"merchandise_available",
        ->setThreedsCardHolderAccount('{"creation_date"=>"16-07-2019", "update_indicator"=>"more_than_60days", "last_change_date"=>"16-04-2020", "password_change_indicator"=>"no_change", "password_change_date"
        ->setThreedsBrowser('{"accept_header"=>"Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser", "java_enabled"=>"true", "language"=>"en-GB", "color_depth"=>"
        ->setThreedsSdk('{"interface"=>"native", "ui_types"=>{"ui_type"=>"multi_select"}, "application_id"=>"fc1650c0-5778-0138-8205-2cbc32a32d65", "encrypted_data"=>"encrypted-data-here", "ephemeral_public_ke

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.recurring.InitRecurringSale3D;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;
import java.net.MalformedURLException;
import java.net.URL;

public class GenesisExample {
  public static void main() throws MalformedURLException {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    InitRecurringSale3D request = new InitRecurringSale3D();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("4901170000000003");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");
    request.setNotificationUrl(new URL("https://www.example.com/notification"));
    request.setReturnSuccessUrl(new URL("http://www.example.com/success"));
    request.setReturnFailureUrl(new URL("http://www.example.com/failure"));


    // Threeds V2 Params
    request.setThreedsControl("{"device_type"=>"browser", "challenge_window_size"=>"full_screen", "challenge_indicator"=>"preference"}");
    request.setThreedsPurchase("{"category"=>"service"}");
    request.setThreedsRecurring("{"expiration_date"=>"16-01-2021", "frequency"=>30}");
    request.setThreedsMerchantRisk("{"shipping_indicator"=>"verified_address", "delivery_timeframe"=>"electronic", "reorder_items_indicator"=>"reordered", "pre_order_purchase_indicator"=>"merchandise_available
    request.setThreedsCardHolderAccount("{"creation_date"=>"16-07-2019", "update_indicator"=>"more_than_60days", "last_change_date"=>"16-04-2020", "password_change_indicator"=>"no_change", "password_change_da
    request.setThreedsBrowser("{"accept_header"=>"Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser", "java_enabled"=>"true", "language"=>"en-GB", "color_depth"
    request.setThreedsSdk("{"interface"=>"native", "ui_types"=>{"ui_type"=>"multi_select"}, "application_id"=>"fc1650c0-5778-0138-8205-2cbc32a32d65", "encrypted_data"=>"encrypted-data-here", "ephemeral_public_

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.init_recurring_sale3d(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "4901170000000003",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "billing_address": {
```

```
      "billing_address": {
        "first_name": "Travis",
        "last_name": "Pastrana",
        "address1": "Muster Str. 12",
        "zip_code": "10178",
        "city": "Los Angeles",
        "state": "CA",
        "country": "US"
      },
      "notification_url": "https://www.example.com/notification",
      "return_success_url": "http://www.example.com/success",
      "return_failure_url": "http://www.example.com/failure",
      "threeds_v2_params": {
        "control": {
          "device_type": "browser",
          "challenge_window_size": "full_screen",
          "challenge_indicator": "preference"
        },
        "purchase": {
          "category": "service"
        },
        "recurring": {
          "expiration_date": "16-01-2021",
          "frequency": 30
        },
        "merchant_risk": {
          "shipping_indicator": "verified_address",
          "delivery_timeframe": "electronic",
          "reorder_items_indicator": "reordered",
          "pre_order_purchase_indicator": "merchandise_available",
          "pre_order_date": "16-08-2020",
          "gift_card": "true",
          "gift_card_count": "2"
        },
        "card_holder_account": {
          "creation_date": "16-07-2019",
          "update_indicator": "more_than_60days",
          "last_change_date": "16-04-2020",
          "password_change_indicator": "no_change",
          "password_change_date": "01-07-2020",
          "shipping_address_usage_indicator": "current_transaction",
          "shipping_address_date_first_used": "11-07-2020",
          "transactions_activity_last_24_hours": "2",
          "transactions_activity_previous_year": "10",
          "provision_attempts_last_24_hours": "1",
          "purchases_count_last_6_months": "5",
          "suspicious_activity_indicator": "no_suspicious_observed",
          "registration_indicator": "30_to_60_days",
          "registration_date": "16-07-2018"
        },
        "browser": {
          "accept_header": "Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser",
          "java_enabled": "true",
          "language": "en-GB",
          "color_depth": "32",
          "screen_height": "900",
          "screen_width": "1440",
          "time_zone_offset": "-120",
          "user_agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.97 Safari/537.36"
        },
        "sdk": {
          "interface": "native",
          "ui_types": {
            "ui_type": "multi_select"
          },
          "application_id": "fc1650c0-5778-0138-8205-2cbc32a32d65",
          "encrypted_data": "encrypted-data-here",
          "ephemeral_public_key_pair": "public-key-pair",
          "max_timeout": "10",
          "reference_number": "sdk-reference-number-here"
        }
      }
    }
  ).send()
    .then(success)
    .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>init_recurring_sale3d</transaction_type>
   <transaction_id>43671</transaction_id>
   <usage>40208 concert tickets</usage>
   <remote_ip>245.253.2.12</remote_ip>
   <amount>100</amount>
   <currency>USD</currency>
   <card_holder>Travis Pastrana</card_holder>
   <card_number>4901170000000003</card_number>
   <expiration_month>12</expiration_month>
   <expiration_year>2021</expiration_year>
   <cvv>834</cvv>
   <customer_email>travis@example.com</customer_email>
   <customer_phone>+1987987987987</customer_phone>
   <billing_address>
     <first_name>Travis</first_name>
     <last_name>Pastrana</last_name>
     <address1>Muster Str. 12</address1>
     <zip_code>10178</zip_code>
     <city>Los Angeles</city>
     <state>CA</state>
     <country>US</country>
   </billing_address>
   <notification_url>https://www.example.com/notification</notification_url>
   <return_success_url>http://www.example.com/success</return_success_url>
   <return_failure_url>http://www.example.com/failure</return_failure_url>
   <threeds_v2_params>
     <control>
       <device_type>browser</device_type>
       <challenge_window_size>full_screen</challenge_window_size>
       <challenge_indicator>preference</challenge_indicator>
     </control>
     <purchase>
       <category>service</category>
     </purchase>
     <recurring>
       <expiration_date>16-01-2021</expiration_date>
       <frequency>30</frequency>
     </recurring>
     <merchant_risk>
       <shipping_indicator>verified_address</shipping_indicator>
       <delivery_timeframe>electronic</delivery_timeframe>
       <reorder_items_indicator>reordered</reorder_items_indicator>
       <pre_order_purchase_indicator>merchandise_available</pre_order_purchase_indicator>
       <pre_order_date>16-08-2020</pre_order_date>
       <gift_card>true</gift_card>
       <gift_card_count>2</gift_card_count>
     </merchant_risk>
     <card_holder_account>
       <creation_date>16-07-2019</creation_date>
       <update_indicator>more_than_60days</update_indicator>
       <last_change_date>16-04-2020</last_change_date>
       <password_change_indicator>no_change</password_change_indicator>
       <password_change_date>01-07-2020</password_change_date>
       <shipping_address_usage_indicator>current_transaction</shipping_address_usage_indicator>
       <shipping_address_date_first_used>11-07-2020</shipping_address_date_first_used>
       <transactions_activity_last_24_hours>2</transactions_activity_last_24_hours>
       <transactions_activity_previous_year>10</transactions_activity_previous_year>
       <provision_attempts_last_24_hours>1</provision_attempts_last_24_hours>
       <purchases_count_last_6_months>5</purchases_count_last_6_months>
       <suspicious_activity_indicator>no_suspicious_observed</suspicious_activity_indicator>
       <registration_indicator>30_to_60_days</registration_indicator>
       <registration_date>16-07-2018</registration_date>
     </card_holder_account>
     <browser>
       <accept_header>Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser</accept_header>
       <java_enabled>true</java_enabled>
       <language>en-GB</language>
       <color_depth>32</color_depth>
       <screen_height>900</screen_height>
       <screen_width>1440</screen_width>
       <time_zone_offset>-120</time_zone_offset>
       <user_agent>Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.97 Safari/537.36</user_agent>
     </browser>
     <sdk>
       <interface>native</interface>
       <ui_types>
         <ui_type>multi_select</ui_type>
       </ui_types>
       <application_id>fc1650c0-5778-0138-8205-2cbc32a32d65</application_id>
       <encrypted_data>encrypted-data-here</encrypted_data>
       <ephemeral_public_key_pair>public-key-pair</ephemeral_public_key_pair>
       <max_timeout>10</max_timeout>
       <reference_number>sdk-reference-number-here</reference_number>
     </sdk>
   </threeds_v2_params>
 </payment_transaction>'
```

Asynchronous 3 D Sv2 Fallback 3dsv1 With 3ds Method Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Recurring\InitRecurringSale3D');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('4901164281364345')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')
        ->setNotificationUrl('https://www.example.com/notification')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')


        // Threeds V2 Params
        ->setThreedsControl('{"device_type"=>"browser", "challenge_window_size"=>"full_screen", "challenge_indicator"=>"preference"}')
        ->setThreedsPurchase('{"category"=>"service"}')
        ->setThreedsRecurring('{"expiration_date"=>"16-01-2021", "frequency"=>30}')
        ->setThreedsMerchantRisk('{"shipping_indicator"=>"verified_address", "delivery_timeframe"=>"electronic", "reorder_items_indicator"=>"reordered", "pre_order_purchase_indicator"=>"merchandise_available",
        ->setThreedsCardHolderAccount('{"creation_date"=>"16-07-2019", "update_indicator"=>"more_than_60days", "last_change_date"=>"16-04-2020", "password_change_indicator"=>"no_change", "password_change_date"
        ->setThreedsBrowser('{"accept_header"=>"Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser", "java_enabled"=>"true", "language"=>"en-GB", "color_depth"=>"
        ->setThreedsSdk('{"interface"=>"native", "ui_types"=>{"ui_type"=>"multi_select"}, "application_id"=>"fc1650c0-5778-0138-8205-2cbc32a32d65", "encrypted_data"=>"encrypted-data-here", "ephemeral_public_ke

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.recurring.InitRecurringSale3D;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;
import java.net.MalformedURLException;
import java.net.URL;

public class GenesisExample {
  public static void main() throws MalformedURLException {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    InitRecurringSale3D request = new InitRecurringSale3D();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("4901164281364345");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");
    request.setNotificationUrl(new URL("https://www.example.com/notification"));
    request.setReturnSuccessUrl(new URL("http://www.example.com/success"));
    request.setReturnFailureUrl(new URL("http://www.example.com/failure"));


    // Threeds V2 Params
    request.setThreedsControl("{"device_type"=>"browser", "challenge_window_size"=>"full_screen", "challenge_indicator"=>"preference"}");
    request.setThreedsPurchase("{"category"=>"service"}");
    request.setThreedsRecurring("{"expiration_date"=>"16-01-2021", "frequency"=>30}");
    request.setThreedsMerchantRisk("{"shipping_indicator"=>"verified_address", "delivery_timeframe"=>"electronic", "reorder_items_indicator"=>"reordered", "pre_order_purchase_indicator"=>"merchandise_available
    request.setThreedsCardHolderAccount("{"creation_date"=>"16-07-2019", "update_indicator"=>"more_than_60days", "last_change_date"=>"16-04-2020", "password_change_indicator"=>"no_change", "password_change_da
    request.setThreedsBrowser("{"accept_header"=>"Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser", "java_enabled"=>"true", "language"=>"en-GB", "color_depth"
    request.setThreedsSdk("{"interface"=>"native", "ui_types"=>{"ui_type"=>"multi_select"}, "application_id"=>"fc1650c0-5778-0138-8205-2cbc32a32d65", "encrypted_data"=>"encrypted-data-here", "ephemeral_public_

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.init_recurring_sale3d(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "4901164281364345",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "billing_address": {
```

```
      "billing_address": {
        "first_name": "Travis",
        "last_name": "Pastrana",
        "address1": "Muster Str. 12",
        "zip_code": "10178",
        "city": "Los Angeles",
        "state": "CA",
        "country": "US"
      },
      "notification_url": "https://www.example.com/notification",
      "return_success_url": "http://www.example.com/success",
      "return_failure_url": "http://www.example.com/failure",
      "threeds_v2_params": {
        "control": {
          "device_type": "browser",
          "challenge_window_size": "full_screen",
          "challenge_indicator": "preference"
        },
        "purchase": {
          "category": "service"
        },
        "recurring": {
          "expiration_date": "16-01-2021",
          "frequency": 30
        },
        "merchant_risk": {
          "shipping_indicator": "verified_address",
          "delivery_timeframe": "electronic",
          "reorder_items_indicator": "reordered",
          "pre_order_purchase_indicator": "merchandise_available",
          "pre_order_date": "16-08-2020",
          "gift_card": "true",
          "gift_card_count": "2"
        },
        "card_holder_account": {
          "creation_date": "16-07-2019",
          "update_indicator": "more_than_60days",
          "last_change_date": "16-04-2020",
          "password_change_indicator": "no_change",
          "password_change_date": "01-07-2020",
          "shipping_address_usage_indicator": "current_transaction",
          "shipping_address_date_first_used": "11-07-2020",
          "transactions_activity_last_24_hours": "2",
          "transactions_activity_previous_year": "10",
          "provision_attempts_last_24_hours": "1",
          "purchases_count_last_6_months": "5",
          "suspicious_activity_indicator": "no_suspicious_observed",
          "registration_indicator": "30_to_60_days",
          "registration_date": "16-07-2018"
        },
        "browser": {
          "accept_header": "Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser",
          "java_enabled": "true",
          "language": "en-GB",
          "color_depth": "32",
          "screen_height": "900",
          "screen_width": "1440",
          "time_zone_offset": "-120",
          "user_agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.97 Safari/537.36"
        },
        "sdk": {
          "interface": "native",
          "ui_types": {
            "ui_type": "multi_select"
          },
          "application_id": "fc1650c0-5778-0138-8205-2cbc32a32d65",
          "encrypted_data": "encrypted-data-here",
          "ephemeral_public_key_pair": "public-key-pair",
          "max_timeout": "10",
          "reference_number": "sdk-reference-number-here"
        }
      }
    }
  ).send()
    .then(success)
    .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>init_recurring_sale3d</transaction_type>
   <transaction_id>43671</transaction_id>
   <usage>40208 concert tickets</usage>
   <remote_ip>245.253.2.12</remote_ip>
   <amount>100</amount>
   <currency>USD</currency>
   <card_holder>Travis Pastrana</card_holder>
   <card_number>4901164281364345</card_number>
   <expiration_month>12</expiration_month>
   <expiration_year>2021</expiration_year>
   <cvv>834</cvv>
   <customer_email>travis@example.com</customer_email>
   <customer_phone>+1987987987987</customer_phone>
   <billing_address>
     <first_name>Travis</first_name>
     <last_name>Pastrana</last_name>
     <address1>Muster Str. 12</address1>
     <zip_code>10178</zip_code>
     <city>Los Angeles</city>
     <state>CA</state>
     <country>US</country>
   </billing_address>
   <notification_url>https://www.example.com/notification</notification_url>
   <return_success_url>http://www.example.com/success</return_success_url>
   <return_failure_url>http://www.example.com/failure</return_failure_url>
   <threeds_v2_params>
     <control>
       <device_type>browser</device_type>
       <challenge_window_size>full_screen</challenge_window_size>
       <challenge_indicator>preference</challenge_indicator>
     </control>
     <purchase>
       <category>service</category>
     </purchase>
     <recurring>
       <expiration_date>16-01-2021</expiration_date>
       <frequency>30</frequency>
     </recurring>
     <merchant_risk>
       <shipping_indicator>verified_address</shipping_indicator>
       <delivery_timeframe>electronic</delivery_timeframe>
       <reorder_items_indicator>reordered</reorder_items_indicator>
       <pre_order_purchase_indicator>merchandise_available</pre_order_purchase_indicator>
       <pre_order_date>16-08-2020</pre_order_date>
       <gift_card>true</gift_card>
       <gift_card_count>2</gift_card_count>
     </merchant_risk>
     <card_holder_account>
       <creation_date>16-07-2019</creation_date>
       <update_indicator>more_than_60days</update_indicator>
       <last_change_date>16-04-2020</last_change_date>
       <password_change_indicator>no_change</password_change_indicator>
       <password_change_date>01-07-2020</password_change_date>
       <shipping_address_usage_indicator>current_transaction</shipping_address_usage_indicator>
       <shipping_address_date_first_used>11-07-2020</shipping_address_date_first_used>
       <transactions_activity_last_24_hours>2</transactions_activity_last_24_hours>
       <transactions_activity_previous_year>10</transactions_activity_previous_year>
       <provision_attempts_last_24_hours>1</provision_attempts_last_24_hours>
       <purchases_count_last_6_months>5</purchases_count_last_6_months>
       <suspicious_activity_indicator>no_suspicious_observed</suspicious_activity_indicator>
       <registration_indicator>30_to_60_days</registration_indicator>
       <registration_date>16-07-2018</registration_date>
     </card_holder_account>
     <browser>
       <accept_header>Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser</accept_header>
       <java_enabled>true</java_enabled>
       <language>en-GB</language>
       <color_depth>32</color_depth>
       <screen_height>900</screen_height>
       <screen_width>1440</screen_width>
       <time_zone_offset>-120</time_zone_offset>
       <user_agent>Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.97 Safari/537.36</user_agent>
     </browser>
     <sdk>
       <interface>native</interface>
       <ui_types>
         <ui_type>multi_select</ui_type>
       </ui_types>
       <application_id>fc1650c0-5778-0138-8205-2cbc32a32d65</application_id>
       <encrypted_data>encrypted-data-here</encrypted_data>
       <ephemeral_public_key_pair>public-key-pair</ephemeral_public_key_pair>
       <max_timeout>10</max_timeout>
       <reference_number>sdk-reference-number-here</reference_number>
     </sdk>
   </threeds_v2_params>
 </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **init_recurring_sale3d** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | required* | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| moto | optional | "true" | Signifies whether a MOTO (mail order telephone order) transaction is performed. Contacttech-support@e-comprocessing.com for more details |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| notification_url | required[1] | url | URL at merchant where gateway sends outcome of transaction. |
| return_success_url | required[1] | url | URL where customer is sent to after successful payment |
| return_failure_url | required[1] | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| card_holder | required | string(255) | Full name of customer as printed on credit card (first name and last name at least) |
| card_number | required | 13 to 16 digits | Complete cc number of customer |
| cvv | required* | 3 to 4 digits | cvv of cc, requirement is based on terminal configuration |
| expiration_month | required | MM | Expiration month as printed on credit card |
| expiration_year | required | YYYY | Expiration year as printed on credit card |
| token | optional | string(36) | See Tokenization for more details. If present, the cardholder parameters can be omitted. Cannot be set together with `remember_card` |
| remember_card | optional | "true" | See Tokenize. Tokenizes cardholder parameters. Cannot be set together with `token` |
| consumer_id | optional | string(10) | See Consumers and Tokenization. Combine with `remember_card` to tokenize or with `token` to use token |
| **credential_on_file** | required* | | See Credential On File (COF) for more details |
| initial_customer_initiated | required* | string(18) | Initial transaction initiated by customer. Required for external tokenization, and optional for gateway-based tokenization |
| subsequent_customer_initiated | required* | string(18) | Subsequent transaction initiated by customer. Required for external tokenization, and optional for gateway-based tokenization |
| merchant_unscheduled | optional | string(20) | Transaction is initiated by the merchant |
| customer_email | required* | e-mail address | Must contain valid e-mail of customer |
| customer_phone | required* | string(32) | Must contain valid phone number of customer |
| birth_date | required* | dd-mm-yyyy | Required when MCC is a Financial Services one (e.g. MCC 6012) and either card brand is Visa or Mastercard/Maestro with UK-based merchant, UK-based bin (domestic), and DEBIT card type |
| document_id | required* | sring(20) | Document ID of the consumer. See Document ID Parameter for more details. |
| fx_rate_id | optional | integer | See Get rates for FX Service. Offers the option to use a specific FX rate to convert the transaction processing amount. Used FX rate should have the same source currency as the processing currency. Contact tech-support@e-comprocessing.com for more details |
| **business_attributes** | required* | | Check business attributes section. |
| event_start_date | required* | dd-mm-yyyy | The date when event starts in format dd-mm-yyyy |
| event_end_date | required* | dd-mm-yyyy | The date when event ends in format dd-mm-yyyy |
| event_organizer_id | required* | string | |
| event_id | required* | string | |
| date_of_order | required* | dd-mm-yyyy | The date when order was placed in format dd-mm-yyyy |
| delivery_date | required* | dd-mm-yyyy | Date of the expected delivery in format dd-mm-yyyy |
| name_of_the_supplier | required* | string | |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |

| Parameter | Required | Format | Description |
|---|---|---|---|
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required* | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |
| **mpi_params** | required[2] | | |
| cavv | required[3] | string(255) | Verification Id of the authentication. Please note this can be the CAVV for Visa Card or UCAF to identify MasterCard. |
| eci | required[3] | string(255) | See Electronic Commerce Indicator as returned from the MPI for details |
| xid | required[3] | string(255) | Transaction ID generated by the 3DSv1 service that uniquely identifies the 3DSecure request. Note, it should not be submitted for transactions, which are though the 3DSv2 authentication protocol. |
| protocol_version | required[4] | string | The used 3DS protocol version. Default is 1 if not supplied. |
| directory_server_id | required[4] | string | The Directory Server ID used for 3DSecure transactions through the 3DSv2 authentication protocol. |
| **dynamic_descriptor_params** | optional | | |
| merchant_name | optional | string(25) | Allows to dynamically override the charge descriptor |
| merchant_city | optional | string(13) | Allows to dynamically override the merchant phone number |
| sub_merchant_id | optional | string(15) | Allows to dynamically override the sub-merchant ID. |
| **sca_params** | optional | | SCA params |
| exemption | optional | string | The exemption that the transaction should take advantage of. |
| visa_merchant_id | required[5] | string(8) | VMID assigned by Visa if participating in Trusted merchant program. |
| **threeds_v2_params** | required* | | 3DSv2 async parameters. They must be submitted in order to use the 3DSv2 authentication protocol in asynchronous workflow |
| **control** | required* | | General params for preferences in authentication flow and providing device interface information. |
| device_type | required* | string | Identifies the device channel of the consumer, **required** in the 3DSv2 authentication protocol. For more information, go to 3DSv2 control params |
| challenge_window_size | required* | string | Identifies the size of the challenge window for the consumer. For more information, go to3DSv2 control params |
| challenge_indicator | optional | string | The value has weight and might impact the decision whether a challenge will be required for the transaction or not. If not provided, it will be interpreted as **no_preference**. For more information, go to3DSv2 control params |
| **purchase** | required* | | Purchase related params providing with additional information regarding the order. |
| category | required* | string | Identifies the type of transaction being authenticated. This field is required in some markets.Accepted values are: **goods**, **service**, **check_acceptance**, **account_funding**, **quasi_cash**, **prepaid_activation**, **loan**. |
| **merchant_risk** | recommended | | Merchant risk assessment params. They are all optional, but recommended' |
| shipping_indicator | optional | string(16) | Indicator code that most accurately describes the shipping method for the cardholder specific transaction. If one or more items are included in the sale, use the Shipping Indicator code for the physical goods. If all digital goods, use the code that describes the most expensive item. Accepted values are: **same_as_billing**, **stored_address**, **verified_address**, **pick_up**, **digital_goods**, **travel**, **event_tickets**, **other**. |
| delivery_timeframe | optional | string(11) | Indicates the merchandise delivery timeframe. Accepted values are: **electronic**, **same_day**, **over_night**, **another_day**. |

| Parameter | Required | Format | Description |
|---|---|---|---|
| reorder_items_indicator | optional | string(10) | Indicates whether the cardholder is reordering previously purchased merchandise. Accepted values are: **first_time**, **reordered**. |
| pre_order_purchase_indicator | optional | string(21) | Indicates whether cardholder is placing an order for merchandise with a future-availability or release date. Accepted values are: **merchandise_available**, **future_availability**. |
| pre_order_date | optional | dd-mm-yyyy | For a pre-ordered purchase, the expected date that the merchandise will be available. |
| gift_card | optional | 'true' | Prepaid or gift card purchase. |
| gift_card_count | optional | integer | For prepaid or gift card purchase, total count of individual prepaid or gift cards/codes purchased. The value is limited to **99**. |
| **card_holder_account** | recommended | | Cardholder account additional information. They are all optional, but recommended, because they have a significant impact on approval rates |
| creation_date | optional | dd-mm-yyyy | Date that the cardholder opened the account with the 3DS Requester. |
| update_indicator | optional | string(19) | Length of time since the cardholder's account information with the 3DS Requestor was last changed. Includes Billing or Shipping address, new payment account, or new user(s) added. Accepted values are: **current_transaction**, **less_than_30days**, **30_to_60_days**, **more_than_60days**. |
| last_change_date | optional | dd-mm-yyyy | Date that the cardholder's account with the 3DS Requestor was last changed. Including Billing or Shipping address, new payment accout, or new user(s) added. |
| password_change_indicator | optional | string(18) | Length of time since the cardholder account with the 3DS Requestor had a password change or account reset. Accepted values are: **no_change**, **during_transaction**, **less_than_30days**, **30_to_60_days**, **more_than_60days**. |
| password_change_date | optional | dd-mm-yyyy | Date that cardholder's account with the 3DS Requestor had a password change or account reset. |
| shipping_address_usage_indicator | optional | string(19) | Indicates when the shipping address used for this transaction was first used with the 3DS Requestor. Accepted values are: **current_transaction**, **less_than_30days**, **30_to_60_days**, **more_than_60days**. |
| shipping_address_date_first_used | optional | dd-mm-yyyy | Date when the shipping address used for this transaction was first used with the 3DS Requestor. |
| transactions_activity_last_24_hours | optional | integer | Number of transactions (successful and abandoned) for this cardholder account with the 3DS Requestor across all payment accounts in the previous 24 hours. |
| transactions_activity_previous_year | optional | integer | Number of transactions (successful and abandoned) for this cardholder account with the 3DS Requestor across all payment accounts in the previous year. |
| provision_attempts_last_24_hours | optional | integer | Number of Add Card attempts in the last 24 hours. |
| purchases_count_last_6_months | optional | integer | Number of purchases with this cardholder account during the previous six months. |
| suspicious_activity_indicator | optional | string(22) | Indicates whether the 3DS Requestor has experienced suspicious activity (including previous fraud) on the cardholder account. Accepted values are: **no_suspicious_observed**, **suspicious_observed**. |
| registration_indicator | optional | string(19) | Indicates the length of time that the payment account was enrolled in the cardholder's account with the 3DS Requester. Accepted values are: **guest_checkout**, **current_transaction**, **less_than_30days**, **30_to_60_days**, **more_than_60days**. |
| registration_date | optional | dd-mm-yyyy | Date that the payment account was enrolled in the cardholder's account with the 3DS Requestor. |
| **browser** | required* | | For browser-based transactions. They are all *required* in case the **device_type** is set to **browser** |
| accept_header | required* | string(2048) | Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser. |
| java_enabled | required* | boolean | Boolean that represents the ability of the cardholder browser to execute Java. The value can be retrieved by accessing a property of the navigator with JavaScript, `navigator.javaEnabled`. |
| language | required* | string(8) | Value representing the browser language as defined in IETF BCP47. The value can be retrieved by accessing a property of the navigator with JavaScript, `navigator.language`. |
| color_depth | required* | integer | Value representing the bit depth of the colour palette for displaying images, in bits per pixel. Obtained from Cardholder browser using the `screen.colorDepth` property. The value can be one of **1**, **4**, **8**, **15**, **16**, **24**, **32**, **48** |
| screen_height | required* | integer | Total height of the Cardholder's screen in pixels. Value is returned from the screen.height property. |
| screen_width | required* | integer | Total width of the Cardholder's screen in pixels. Value is returned from the screen.width property. |
| time_zone_offset | required* | string(5) | Time difference between UTC time and the Cardholder browser local time, in minutes. The value can be retrieved using Javascript `getTimezoneOffset()` method. |
| user_agent | required* | string(2048) | Exact content of the HTTP user-agent header. |
| **sdk** | required* | | For application-based transactions. They are all *required* in case the **device_type** is set to **application** |
| interface | required* | string(6) | SDK Interface types that the device of the consumer supports for displaying specific challenge interfaces within the SDK. Accepted values are: **native**, **html**, **both**. |
| **ui_types** | required* | | Lists all UI types that the device of the consumer supports for displaying specific challenge interfaces within the SDK. |

| Parameter | Required | Format | Description |
|---|---|---|---|
| ui_type | required* | string(13) | UI type that the device of the consumer supports for displaying specific challenge interface. Accepted values are: **text**, **single_select**, **multi_select**, **out_of_bag**, **other_html**. |
| application_id | required* | string(36) | Universally unique ID created upon all installations and updates of the 3DS Requestor APp on a Customer Device. This will be newly generated and stored by the 3DS SDK for each installation or update. The field is limited to 36 characters and it shall have a canonical format as defined in IETF RFC 4122. |
| encrypted_data | required* | string(64000) | JWE Object as defined Section 6.2.2.1 containing data encrypted by the SDK for the DS to decrypt. The data will be present when sending to DS, but not present from DS to ACS. |
| ephemeral_public_key_pair | required* | string(256) | Public key component of the ephemeral key pair generated by the 3DS SDK and used to establish session keys between the 3DS SDK and ACS. In AReq, this data element is contained within the ACS Signed Content JWS Object. The field is limited to maximum 256 characters. |
| max_timeout | required* | integer | Indicates the maximum amount of time (in minutes) for all exchanges. The field shall have value greater or equals than 05. |
| reference_number | required* | string(32) | Identifies the vendor and version of the 3DS SDK that is integrated in a 3DS Requestor App, assigned by EMVCo when the 3DS SDK is approved. The field is limited to 32 characters. |
| **recurring** | required* | | Additional recurring details. |
| expiration_date | required* | dd-mm-yyyy | A future date indicating the end date for any further subsequent transactions. For more information, go to 3DSv2 recurring params |
| frequency | required* | integer | Indicates the minimum number of days between subsequent transactions. For more information, go to3DSv2 recurring params |

`required* = conditionally required`

`1 - required if mpi_params is not present, the transaction will be handled asynchronously. Not required if configured on Terminal or Merchant level. Contact [tech-support@#{email_domain_name}](mailto:tech-support@#{email_domain_name}) for more details.`

`2 - required if transaction should be handled synchronous.`

`3 - eci is always required if mpi_params is present. xid and cavv are not required for the 3D attempted only workflow, where the merchant does not have the xid/cavv, only the eci.`

`4 - protocol_version is only required if 3DSv2 is used. When missing 3DSv1 will be used. directory_server_id is mandatory when protocol_version is 2.`

`5 - visa_merchant_id is required when exemption value is trusted_merchant.`

Successful Synchronous Response

```
stdClass Object
(
    [transaction_type] => init_recurring_sale3d
    [status] => approved
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [consumer_id] => 123456
    [token] => ee946db8-d7db-4bb7-b608-b65b153e127d
    [avs_response_code] => 5I
    [avs_response_text] => Response provided by issuer processor; Address information not verified
    [authorization_code] => 345678
    [retrieval_reference_number] => 016813015184
    [response_code] => 00
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:17.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
    [partial_approval] => true
    [scheme_transaction_identifier] => 019091214161031
)
```

```xml
<payment_response content=[
    <transaction_type content=[init_recurring_sale3d]>
    <status content=[approved]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <consumer_id content=[123456]>
    <token content=[ee946db8-d7db-4bb7-b608-b65b153e127d]>
    <avs_response_code content=[5I]>
    <avs_response_text content=[Response provided by issuer processor; Address information not verified]>
    <authorization_code content=[345678]>
    <retrieval_reference_number content=[016813015184]>
    <response_code content=[00]>
    <timestamp content=[2020-07-15T12:08:17Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[true]>
    <partial_approval content=[true]>
    <scheme_transaction_identifier content=[019091214161031]>
]>
```

```json
{
    transaction_type: "init_recurring_sale3d",
    status: "approved",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    consumer_id: "123456",
    token: "ee946db8-d7db-4bb7-b608-b65b153e127d",
    avs_response_code: "5I",
    avs_response_text: "Response provided by issuer processor; Address information not verified",
    authorization_code: "345678",
    retrieval_reference_number: "016813015184",
    response_code: "00",
    timestamp: "2020-07-15T12:08:17Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "true",
    partial_approval: "true",
    scheme_transaction_identifier: "019091214161031",
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>init_recurring_sale3d</transaction_type>
  <status>approved</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <consumer_id>123456</consumer_id>
  <token>ee946db8-d7db-4bb7-b608-b65b153e127d</token>
  <avs_response_code>5I</avs_response_code>
  <avs_response_text>Response provided by issuer processor; Address information not verified</avs_response_text>
  <authorization_code>345678</authorization_code>
  <retrieval_reference_number>016813015184</retrieval_reference_number>
  <response_code>00</response_code>
  <timestamp>2020-07-15T12:08:17Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
  <partial_approval>true</partial_approval>
  <scheme_transaction_identifier>019091214161031</scheme_transaction_identifier>
</payment_transaction>
```


Successful Asynchronous Response

```
stdClass Object
(
    [transaction_type] => init_recurring_sale3d
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [consumer_id] => 123456
    [token] => ee946db8-d7db-4bb7-b608-b65b153e127d
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/threeds/authentication/fd3bbd0509d8ac3ef0708b76220b2509
    [redirect_url_type] => 3ds_v1_payer_authentication
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:17.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
    [scheme_transaction_identifier] => 019091214161031
)
```

```
<payment_response content=[
    <transaction_type content=[init_recurring_sale3d]>
    <status content=[pending_async]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <consumer_id content=[123456]>
    <token content=[ee946db8-d7db-4bb7-b608-b65b153e127d]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <redirect_url content=[https://staging.gate.e-comprocessing.net/threeds/authentication/fd3bbd0509d8ac3ef0708b76220b2509]>
    <redirect_url_type content=[3ds_v1_payer_authentication]>
    <timestamp content=[2020-07-15T12:08:17Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[true]>
    <scheme_transaction_identifier content=[019091214161031]>
]>
```

```
{
    transaction_type: "init_recurring_sale3d",
    status: "pending_async",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    consumer_id: "123456",
    token: "ee946db8-d7db-4bb7-b608-b65b153e127d",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    redirect_url: "https://staging.gate.e-comprocessing.net/threeds/authentication/fd3bbd0509d8ac3ef0708b76220b2509",
    redirect_url_type: "3ds_v1_payer_authentication",
    timestamp: "2020-07-15T12:08:17Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "true",
    scheme_transaction_identifier: "019091214161031",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>init_recurring_sale3d</transaction_type>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <consumer_id>123456</consumer_id>
  <token>ee946db8-d7db-4bb7-b608-b65b153e127d</token>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <redirect_url>https://staging.gate.e-comprocessing.net/threeds/authentication/fd3bbd0509d8ac3ef0708b76220b2509</redirect_url>
  <redirect_url_type>3ds_v1_payer_authentication</redirect_url_type>
  <timestamp>2020-07-15T12:08:17Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
  <scheme_transaction_identifier>019091214161031</scheme_transaction_identifier>
</payment_transaction>
```


Frictionless / Challenge With Synchronous 3ds Method Response

```
stdClass Object
(
    [transaction_type] => init_recurring_sale3d
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [consumer_id] => 123456
    [token] => ee946db8-d7db-4bb7-b608-b65b153e127d
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/threeds/threeds_method/1a096da019a4679a828baa175a306ae6
    [redirect_url_type] => 3ds_v2_threeds_method
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:17.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
    [scheme_transaction_identifier] => 019091214161031
    [authorization_code] => 345678
    [retrieval_reference_number] => 016813015184
)
```

```
<payment_response content=[
    <transaction_type content=[init_recurring_sale3d]>
    <status content=[pending_async]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <consumer_id content=[123456]>
    <token content=[ee946db8-d7db-4bb7-b608-b65b153e127d]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <redirect_url content=[https://staging.gate.e-comprocessing.net/threeds/threeds_method/1a096da019a4679a828baa175a306ae6]>
    <redirect_url_type content=[3ds_v2_threeds_method]>
    <timestamp content=[2020-07-15T12:08:17Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[true]>
    <scheme_transaction_identifier content=[019091214161031]>
    <authorization_code content=[345678]>
    <retrieval_reference_number content=[016813015184]>
]>
```

```
{
    transaction_type: "init_recurring_sale3d",
    status: "pending_async",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    consumer_id: "123456",
    token: "ee946db8-d7db-4bb7-b608-b65b153e127d",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    redirect_url: "https://staging.gate.e-comprocessing.net/threeds/threeds_method/1a096da019a4679a828baa175a306ae6",
    redirect_url_type: "3ds_v2_threeds_method",
    timestamp: "2020-07-15T12:08:17Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "true",
    scheme_transaction_identifier: "019091214161031",
    authorization_code: "345678",
    retrieval_reference_number: "016813015184",
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>init_recurring_sale3d</transaction_type>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <consumer_id>123456</consumer_id>
  <token>ee946db8-d7db-4bb7-b608-b65b153e127d</token>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <redirect_url>https://staging.gate.e-comprocessing.net/threeds/threeds_method/1a096da019a4679a828baa175a306ae6</redirect_url>
  <redirect_url_type>3ds_v2_threeds_method</redirect_url_type>
  <timestamp>2020-07-15T12:08:17Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
  <scheme_transaction_identifier>019091214161031</scheme_transaction_identifier>
  <authorization_code>345678</authorization_code>
  <retrieval_reference_number>016813015184</retrieval_reference_number>
</payment_transaction>
```


Frictionless / Challenge With Asynchronous 3ds Method Response


```
stdClass Object
(
    [transaction_type] => init_recurring_sale3d
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [consumer_id] => 123456
    [token] => ee946db8-d7db-4bb7-b608-b65b153e127d
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/threeds/threeds_method/1a096da019a4679a828baa175a306ae6/continue
    [redirect_url_type] => 3ds_v2_threeds_method
    [threeds_method_url] => https://staging.gate.e-comprocessing.net/threeds/threeds_method/649e1ff35c61
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:17.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
    [scheme_transaction_identifier] => 019091214161031
    [authorization_code] => 345678
    [retrieval_reference_number] => 016813015184
)
```



```
<payment_response content=[
    <transaction_type content=[init_recurring_sale3d]>
    <status content=[pending_async]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <consumer_id content=[123456]>
    <token content=[ee946db8-d7db-4bb7-b608-b65b153e127d]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <redirect_url content=[https://staging.gate.e-comprocessing.net/threeds/threeds_method/1a096da019a4679a828baa175a306ae6/continue]>
    <redirect_url_type content=[3ds_v2_threeds_method]>
    <threeds_method_url content=[https://staging.gate.e-comprocessing.net/threeds/threeds_method/649e1ff35c61]>
    <timestamp content=[2020-07-15T12:08:17Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[true]>
    <scheme_transaction_identifier content=[019091214161031]>
    <authorization_code content=[345678]>
    <retrieval_reference_number content=[016813015184]>
]>
```

```
{
    transaction_type: "init_recurring_sale3d",
    status: "pending_async",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    consumer_id: "123456",
    token: "ee946db8-d7db-4bb7-b608-b65b153e127d",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    redirect_url: "https://staging.gate.e-comprocessing.net/threeds/threeds_method/1a096da019a4679a828baa175a306ae6/continue",
    redirect_url_type: "3ds_v2_threeds_method",
    threeds_method_url: "https://staging.gate.e-comprocessing.net/threeds/threeds_method/649e1ff35c61",
    timestamp: "2020-07-15T12:08:17Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "true",
    scheme_transaction_identifier: "019091214161031",
    authorization_code: "345678",
    retrieval_reference_number: "016813015184",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>init_recurring_sale3d</transaction_type>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <consumer_id>123456</consumer_id>
  <token>ee946db8-d7db-4bb7-b608-b65b153e127d</token>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <redirect_url>https://staging.gate.e-comprocessing.net/threeds/threeds_method/1a096da019a4679a828baa175a306ae6/continue</redirect_url>
  <redirect_url_type>3ds_v2_threeds_method</redirect_url_type>
  <threeds_method_url>https://staging.gate.e-comprocessing.net/threeds/threeds_method/649e1ff35c61</threeds_method_url>
  <timestamp>2020-07-15T12:08:17Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
  <scheme_transaction_identifier>019091214161031</scheme_transaction_identifier>
  <authorization_code>345678</authorization_code>
  <retrieval_reference_number>016813015184</retrieval_reference_number>
</payment_transaction>
```


Challenge Without 3 Ds Method Response


```
stdClass Object
(
    [transaction_type] => init_recurring_sale3d
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [consumer_id] => 123456
    [token] => ee946db8-d7db-4bb7-b608-b65b153e127d
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/threeds/authentication/fd3bbd0509d8ac3ef0708b76220b2509
    [redirect_url_type] => 3ds_v2_challenge
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:17.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
    [scheme_transaction_identifier] => 019091214161031
)
```

```
<payment_response content=[
    <transaction_type content=[init_recurring_sale3d]>
    <status content=[pending_async]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <consumer_id content=[123456]>
    <token content=[ee946db8-d7db-4bb7-b608-b65b153e127d]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <redirect_url content=[https://staging.gate.e-comprocessing.net/threeds/authentication/fd3bbd0509d8ac3ef0708b76220b2509]>
    <redirect_url_type content=[3ds_v2_challenge]>
    <timestamp content=[2020-07-15T12:08:17Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[true]>
    <scheme_transaction_identifier content=[019091214161031]>
]>
```

```
{
    transaction_type: "init_recurring_sale3d",
    status: "pending_async",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    consumer_id: "123456",
    token: "ee946db8-d7db-4bb7-b608-b65b153e127d",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    redirect_url: "https://staging.gate.e-comprocessing.net/threeds/authentication/fd3bbd0509d8ac3ef0708b76220b2509",
    redirect_url_type: "3ds_v2_challenge",
    timestamp: "2020-07-15T12:08:17Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "true",
    scheme_transaction_identifier: "019091214161031",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>init_recurring_sale3d</transaction_type>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <consumer_id>123456</consumer_id>
  <token>ee946db8-d7db-4bb7-b608-b65b153e127d</token>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <redirect_url>https://staging.gate.e-comprocessing.net/threeds/authentication/fd3bbd0509d8ac3ef0708b76220b2509</redirect_url>
  <redirect_url_type>3ds_v2_challenge</redirect_url_type>
  <timestamp>2020-07-15T12:08:17Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
  <scheme_transaction_identifier>019091214161031</scheme_transaction_identifier>
</payment_transaction>
```


Fallback To 3ds1 Without 3ds Method Response


```
stdClass Object
(
    [transaction_type] => init_recurring_sale3d
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [consumer_id] => 123456
    [token] => ee946db8-d7db-4bb7-b608-b65b153e127d
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/threeds/authentication/fd3bbd0509d8ac3ef0708b76220b2509
    [redirect_url_type] => 3ds_v1_payer_authentication
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:17.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
    [scheme_transaction_identifier] => 019091214161031
)
```

```xml
<payment_response content=[
    <transaction_type content=[init_recurring_sale3d]>
    <status content=[pending_async]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <consumer_id content=[123456]>
    <token content=[ee946db8-d7db-4bb7-b608-b65b153e127d]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <redirect_url content=[https://staging.gate.e-comprocessing.net/threeds/authentication/fd3bbd0509d8ac3ef0708b76220b2509]>
    <redirect_url_type content=[3ds_v1_payer_authentication]>
    <timestamp content=[2020-07-15T12:08:17Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[true]>
    <scheme_transaction_identifier content=[019091214161031]>
]>
```

```json
{
    transaction_type: "init_recurring_sale3d",
    status: "pending_async",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    consumer_id: "123456",
    token: "ee946db8-d7db-4bb7-b608-b65b153e127d",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    redirect_url: "https://staging.gate.e-comprocessing.net/threeds/authentication/fd3bbd0509d8ac3ef0708b76220b2509",
    redirect_url_type: "3ds_v1_payer_authentication",
    timestamp: "2020-07-15T12:08:17Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "true",
    scheme_transaction_identifier: "019091214161031",
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>init_recurring_sale3d</transaction_type>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <consumer_id>123456</consumer_id>
  <token>ee946db8-d7db-4bb7-b608-b65b153e127d</token>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <redirect_url>https://staging.gate.e-comprocessing.net/threeds/authentication/fd3bbd0509d8ac3ef0708b76220b2509</redirect_url>
  <redirect_url_type>3ds_v1_payer_authentication</redirect_url_type>
  <timestamp>2020-07-15T12:08:17Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
  <scheme_transaction_identifier>019091214161031</scheme_transaction_identifier>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| consumer_id | string(10) | Consumer unique reference. See Consumers |
| token | string(36) | Plain-text token value. See Tokenization |
| gaming | string(255) | Signifies whether a gaming transaction is performed. Gaming transactions usually use MCC 7995, contact tech support for more details. |
| moto | string(255) | Signifies whether a MOTO (mail order telephone order) transaction is performed. Contact tech support for more details. |
| avs_response_code | string(255) | Generated by the card network on trying to match the billing address when performing the address verification. Optional, returned if config is enabled and acquirer supports it. Check AVS Status Codes for details. |
| avs_response_text | string(255) | Gives the human response text for the AVS response code above. Optional, returned if config is enabled and acquirer supports it. Check AVS Status Codes for details. |
| authorization_code | string(6) | Generated by the card network when an authorisation has occurred, used to identify that auth. Consists of 6 alphanumeric chars |
| retrieval_reference_number | string(255) | A reference number used for tracking all messages related to a given cardholder transaction returned by some acquirers. |

| Parameter | Type | Description |
|---|---|---|
| response_code | string(2) | Defines the issuer result of a transaction, the status of a message or some action taken or required. See issuer response codes for details |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| redirect_url | url | URL where the consumer has to be redirected to complete payment process unless a 3DSecure Method is required. For more information, to go 3DSv2 authentication flows |
| redirect_url_type | string(64) | The type of the redirect URL. It might be present only for asynchronous 3D transactions For more information, to go 3DSv2 authentication flows |
| threeds_method_url | url | 3DSecure Method URL. It will be present only then asynchronous 3DSecure Method is required for 3D transaction. For more information, to go 3DSv2 authentication flows |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08- 30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| partial_approval | string(4) | Optional, set to 'true' if partial approval happened. Partially approved amount is then in the amount field. Check Partial Approvals for details |
| sent_to_acquirer | string(255) | "true" or "false" |
| dynamic_descriptor_params | section | Optional, returned only if dynamic desc params are submitted on the API. Note here that the formatted dyn desc params are returned - as they would be submitted to the schemes for settlement. |
| scheme_transaction_identifier | string(32) | Id defined by card schemes |

Error Response

```
stdClass Object
(
    [transaction_type] => init_recurring_sale3d
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [response_code] => 57
    [code] => 340
    [technical_message] => expiration_year is invalid
    [message] => expiration_year is invalid
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:17.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => false
)
```

```
<payment_response content=[
    <transaction_type content=[init_recurring_sale3d]>
    <status content=[error]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <response_code content=[57]>
    <code content=[340]>
    <technical_message content=[expiration_year is invalid]>
    <message content=[expiration_year is invalid]>
    <timestamp content=[2020-07-15T12:08:17Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[false]>
]>
```

```
{
    transaction_type: "init_recurring_sale3d",
    status: "error",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    response_code: "57",
    code: "340",
    technical_message: "expiration_year is invalid",
    message: "expiration_year is invalid",
    timestamp: "2020-07-15T12:08:17Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "false",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>init_recurring_sale3d</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <response_code>57</response_code>
  <code>340</code>
  <technical_message>expiration_year is invalid</technical_message>
  <message>expiration_year is invalid</message>
  <timestamp>2020-07-15T12:08:17Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>false</sent_to_acquirer>
</payment_transaction>
```


**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| moto | string(255) | Signifies whether a MOTO (mail order telephone order) transaction is performed. Contact tech support for more details. |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08- 30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| dynamic_descriptor_params | section | Optional, returned only if dynamic desc params are submitted on the API. Note here that the formatted dyn desc params are returned - as they would be submitted to the schemes for settlement. |

#### NOTIFICATION FOR ASYNCHRONOUS PAYMENTS

When using asynchronous payments like 3-D secured transactions, the transaction depends on user input. Therefore, the outcome of the transaction is not available immediately after sending the request.

The gateway will send a notification to the merchant's server as soon as the payment has been completed (either approved or declined).

If the user cancels the payment (or the user doesn't complete the payment within the given time frame, e.g. 2 hours), the transaction will time out and a notification will be sent.

The notification will be sent as an HTTP POST to the `notification_url` specified in the transaction request XML. See Notifications for the HTTP POST-Data and format. Notification will be pure HTTP or HTTPS-based, depending on the URL given by the merchant in the `notification_url`. In case it is a HTTPS-based notification, no SSL verification of the merchant SSL certificate will be performed. Until a notification echo is rendered by the merchant, there will be up to 10 notifications sent, each with a timeout of 15 minutes. Note that notifications in the PROD env will be sent from 80.82.206.228 and 5.179.225.28 for the PROD env, and only 80.82.206.228 for the STG env, so make sure these IP addresses are whitelisted respectively. Note also that notifications are delivered to either port 80 (HTTP notification) or port 443 (HTTPS notification), so make sure the correct `notification_url` is passed in the transaction request in the first place.

Also see 3-D secure workflow.

# Wallets

**NETELLER**

Neteller transactions are only synchronous. When the payment reaches a final state Genesis gateway sends notification to merchant on the configured url into its account.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Wallets\Neteller');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setNotificationUrl('https://www.example.com/notification')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCustomerAccount('453501020503')
        ->setAccountPassword('908379')
        ->setCustomerEmail('travis@example.com')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.wallets.NetellerRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;
import java.net.MalformedURLException;
import java.net.URL;

public class GenesisExample {
  public static void main() throws MalformedURLException {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    NetellerRequest request = new NetellerRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setNotificationUrl(new URL("https://www.example.com/notification"));
    request.setReturnSuccessUrl(new URL("http://www.example.com/success"));
    request.setReturnFailureUrl(new URL("http://www.example.com/failure"));
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCustomerAccount("453501020503");
    request.setAccountPassword("908379");
    request.setCustomerEmail("travis@example.com");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.neteller(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "notification_url": "https://www.example.com/notification",
    "return_success_url": "http://www.example.com/success",
    "return_failure_url": "http://www.example.com/failure",
    "amount": "100",
    "currency": "USD",
    "customer_account": "453501020503",
    "account_password": "908379",
    "customer_email": "travis@example.com",
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>neteller</transaction_type>
   <transaction_id>43671</transaction_id>
   <usage>40208 concert tickets</usage>
   <remote_ip>245.253.2.12</remote_ip>
   <notification_url>https://www.example.com/notification</notification_url>
   <return_success_url>http://www.example.com/success</return_success_url>
   <return_failure_url>http://www.example.com/failure</return_failure_url>
   <amount>100</amount>
   <currency>USD</currency>
   <customer_account>453501020503</customer_account>
   <account_password>908379</account_password>
   <customer_email>travis@example.com</customer_email>
   <billing_address>
     <first_name>Travis</first_name>
     <last_name>Pastrana</last_name>
     <address1>Muster Str. 12</address1>
     <zip_code>10178</zip_code>
     <city>Los Angeles</city>
     <state>CA</state>
     <country>US</country>
   </billing_address>
 </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **neteller** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| customer_email | required | e-mail address | Must contain valid e-mail of customer |
| customer_phone | required* | string(32) | Must contain valid phone number of customer |
| customer_account | required | string(100) | Customer email/id of neteller account |
| account_password | required | string(6) | Account secret password |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |

| Parameter | Required | Format | Description |
|---|---|---|---|
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

Successful Response

```
stdClass Object
(
    [transaction_type] => neteller
    [status] => approved
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:17.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[neteller]>
    <status content=[approved]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <timestamp content=[2020-07-15T12:08:17Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[true]>
]>
```

```
{
    transaction_type: "neteller",
    status: "approved",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    timestamp: "2020-07-15T12:08:17Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "true",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>neteller</transaction_type>
  <status>approved</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <timestamp>2020-07-15T12:08:17Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |

| Parameter | Type | Description |
|---|---|---|
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [transaction_type] => neteller
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 110
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:17.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => false
)
```

```
<payment_response content=[
    <transaction_type content=[neteller]>
    <status content=[error]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <code content=[110]>
    <message content=[Something went wrong, please contact support!]>
    <timestamp content=[2020-07-15T12:08:17Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[false]>
]>
```

```
{
    transaction_type: "neteller",
    status: "error",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    code: "110",
    message: "Something went wrong, please contact support!",
    timestamp: "2020-07-15T12:08:17Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "false",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>neteller</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>110</code>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:17Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>false</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

**PAYPAL EXPRESS CHECKOUT**

Express Checkout is a fast, easy way for buyers to pay with PayPal. Express Checkout eliminates one of the major causes of checkout abandonment by giving buyers all the transaction details at once, including order details, shipping options, insurance choices, and tax totals.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Alternatives\PaypalExpress');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setNotificationUrl('https://www.example.com/notification')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCustomerEmail('travis@example.com')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.apm.PayPalExpressRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;
import java.net.MalformedURLException;
import java.net.URL;

public class GenesisExample {
  public static void main() throws MalformedURLException {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    PayPalExpressRequest request = new PayPalExpressRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setNotificationUrl(new URL("https://www.example.com/notification"));
    request.setReturnSuccessUrl(new URL("http://www.example.com/success"));
    request.setReturnFailureUrl(new URL("http://www.example.com/failure"));
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCustomerEmail("travis@example.com");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.paypal_express(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "notification_url": "https://www.example.com/notification",
    "return_success_url": "http://www.example.com/success",
    "return_failure_url": "http://www.example.com/failure",
    "amount": "100",
    "currency": "USD",
    "customer_email": "travis@example.com",
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>paypal_express</transaction_type>
   <transaction_id>43671</transaction_id>
   <usage>40208 concert tickets</usage>
   <remote_ip>245.253.2.12</remote_ip>
   <notification_url>https://www.example.com/notification</notification_url>
   <return_success_url>http://www.example.com/success</return_success_url>
   <return_failure_url>http://www.example.com/failure</return_failure_url>
   <amount>100</amount>
   <currency>USD</currency>
   <customer_email>travis@example.com</customer_email>
   <billing_address>
     <first_name>Travis</first_name>
     <last_name>Pastrana</last_name>
     <address1>Muster Str. 12</address1>
     <zip_code>10178</zip_code>
     <city>Los Angeles</city>
     <state>CA</state>
     <country>US</country>
   </billing_address>
 </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **paypal_express** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| customer_email | required* | e-mail address | Must contain valid e-mail of customer |
| customer_phone | required* | string(32) | Must contain valid phone number of customer |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

Successful Response

```
stdClass Object
(
    [transaction_type] => paypal_express
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:17.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[paypal_express]>
    <status content=[pending_async]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <redirect_url content=[https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61]>
    <timestamp content=[2020-07-15T12:08:17Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[true]>
]>
```

```
{
    transaction_type: "paypal_express",
    status: "pending_async",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    redirect_url: "https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61",
    timestamp: "2020-07-15T12:08:17Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "true",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>paypal_express</transaction_type>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
  <timestamp>2020-07-15T12:08:17Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |

| Parameter | Type | Description |
|---|---|---|
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [transaction_type] => paypal_express
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 110
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:17.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => false
)
```

```
<payment_response content=[
    <transaction_type content=[paypal_express]>
    <status content=[error]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <code content=[110]>
    <message content=[Something went wrong, please contact support!]>
    <timestamp content=[2020-07-15T12:08:17Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[false]>
]>
```

```
{
    transaction_type: "paypal_express",
    status: "error",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    code: "110",
    message: "Something went wrong, please contact support!",
    timestamp: "2020-07-15T12:08:17Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "false",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>paypal_express</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>110</code>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:17Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>false</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |

| Parameter | Type | Description |
|---|---|---|
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

#### QIWI

QIWI Wallet is a very popular Eastern European e-wallet.

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>qiwi</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <return_success_url>http://www.example.com/success</return_success_url>
    <return_failure_url>http://www.example.com/failure</return_failure_url>
    <amount>5000</amount>
    <currency>RUB</currency>
    <customer_phone>71234567891</customer_phone>
    <billing_address>
      <first_name>Barney</first_name>
      <last_name>Rubble</last_name>
      <address1>14, Nerazdelni str</address1>
      <zip_code>1407</zip_code>
      <city>Moscow</city>
      <country>RU</country>
    </billing_address>
    <risk_params>
      <user_id>123456</user_id>
    </risk_params>
  </payment_transaction>'
```

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Alternatives\PPRO');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setPaymentType('qiwi')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setAmount('5000')
        ->setCurrency('RUB')
        ->setCustomerPhone('71234567891')


        // Billing Address
        ->setBillingFirstName('Barney')
        ->setBillingLastName('Rubble')
        ->setBillingAddress1('14, Nerazdelni str')
        ->setBillingZipCode('1407')
        ->setBillingCity('Moscow')
        ->setBillingCountry('RU')


        // Risk Params
        ->setRiskUserId('123456');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.apm.PProRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;
import java.net.MalformedURLException;
import java.net.URL;

public class GenesisExample {
  public static void main() throws MalformedURLException {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    PProRequest request = new PProRequest();

    request.setTransactionId("43671");
    request.setPaymentType("qiwi");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setReturnSuccessUrl(new URL("http://www.example.com/success"));
    request.setReturnFailureUrl(new URL("http://www.example.com/failure"));
    request.setAmount(new BigDecimal(5000));
    request.setCurrency("RUB");
    request.setCustomerPhone("71234567891");


    // Billing Address
    request.setBillingFirstname("Barney");
    request.setBillingLastname("Rubble");
    request.setBillingPrimaryAddress("14, Nerazdelni str");
    request.setBillingZipCode("1407");
    request.setBillingCity("Moscow");
    request.setBillingCountry("RU");


    // Risk Params
    request.setRiskUserId("123456");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.ppro(
  {
    "transaction_id": "43671",
    "payment_type": "qiwi",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "return_success_url": "http://www.example.com/success",
    "return_failure_url": "http://www.example.com/failure",
    "amount": "5000",
    "currency": "RUB",
    "customer_phone": "71234567891",
    "billing_address": {
      "first_name": "Barney",
      "last_name": "Rubble",
      "address1": "14, Nerazdelni str",
      "zip_code": "1407",
      "city": "Moscow",
      "country": "RU"
    },
    "risk_params": {
      "user_id": "123456"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>ppro</transaction_type>
   <transaction_id>43671</transaction_id>
   <payment_type>qiwi</payment_type>
   <usage>40208 concert tickets</usage>
   <remote_ip>245.253.2.12</remote_ip>
   <return_success_url>http://www.example.com/success</return_success_url>
   <return_failure_url>http://www.example.com/failure</return_failure_url>
   <amount>5000</amount>
   <currency>RUB</currency>
   <customer_phone>71234567891</customer_phone>
   <billing_address>
     <first_name>Barney</first_name>
     <last_name>Rubble</last_name>
     <address1>14, Nerazdelni str</address1>
     <zip_code>1407</zip_code>
     <city>Moscow</city>
     <country>RU</country>
   </billing_address>
   <risk_params>
     <user_id>123456</user_id>
   </risk_params>
 </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | **ppro** or **qiwi**. Contact tech support at tech-support@e-comprocessing.com for more details. |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| payment_type | required[1] | qiwi | QIWI. Contact tech support at tech-support@e-comprocessing.com for more details. |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| customer_phone | required | string(11) | Must be a 10 digits Russian valid mobile number(either with or without international prefix +7), or International mobile phone number, starting with + followed by at least 9 digits. |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |

| Parameter | Required | Format | Description |
|---|---|---|---|
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

`1 - payment_type must be submitted only when the transaction type is set to ppro`

**Supported countries and countries**

| Country code | Currency code |
|---|---|
| RU | EUR, RUB |
| KZ | EUR, KZT |
| UA | USD |

Successful Response

```
stdClass Object
(
    [transaction_type] => ppro
    [status] => approved
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:17.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 5000
    [currency] => RUB
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[ppro]>
    <status content=[approved]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <timestamp content=[2020-07-15T12:08:17Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[5000]>
    <currency content=[RUB]>
    <sent_to_acquirer content=[true]>
]>
```

```
{
    transaction_type: "ppro",
    status: "approved",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    timestamp: "2020-07-15T12:08:17Z",
    descriptor: "Descriptor one",
    amount: "5000",
    currency: "RUB",
    sent_to_acquirer: "true",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>ppro</transaction_type>
  <status>approved</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <timestamp>2020-07-15T12:08:17Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>5000</amount>
  <currency>RUB</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [transaction_type] => ppro
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 110
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:17.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 5000
    [currency] => RUB
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[ppro]>
    <status content=[error]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <code content=[110]>
    <message content=[Something went wrong, please contact support!]>
    <timestamp content=[2020-07-15T12:08:17Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[5000]>
    <currency content=[RUB]>
    <sent_to_acquirer content=[true]>
]>
```

```
{
    transaction_type: "ppro",
    status: "error",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    code: "110",
    message: "Something went wrong, please contact support!",
    timestamp: "2020-07-15T12:08:17Z",
    descriptor: "Descriptor one",
    amount: "5000",
    currency: "RUB",
    sent_to_acquirer: "true",
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>ppro</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>110</code>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:17Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>5000</amount>
  <currency>RUB</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

#### WEBMONEY

WebMoney is a global settlement system and environment for online business activities. Over 28 million people from all over the world have joined the system. Although WebMoney is targeted mainly at clients in Russia and the Former Soviet Union, it is now used worldwide.

WebMoney transactions can be either asynchronous or synchronous, based on 'is_payout' flag. After a successful validation of transaction parameters, transaction status is set to pending_async, the user is redirected to WebMoney authentication page where he enters additional information to finish the payment. When the payment reaches a final state Genesis gateway sends notification to merchant on the configured url into its account. When the payment is checked as payout then the transaction is synchronous and transaction status is set immediately after the response.

Auth Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Wallets\WebMoney');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setCustomerPhone('+1987987987987')
        ->setCustomerEmail('travis@example.com')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.wallets.WebMoneyRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;
import java.net.MalformedURLException;
import java.net.URL;

public class GenesisExample {
  public static void main() throws MalformedURLException {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    WebMoneyRequest request = new WebMoneyRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setReturnSuccessUrl(new URL("http://www.example.com/success"));
    request.setReturnFailureUrl(new URL("http://www.example.com/failure"));
    request.setCustomerPhone("+1987987987987");
    request.setCustomerEmail("travis@example.com");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.webmoney(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "return_success_url": "http://www.example.com/success",
    "return_failure_url": "http://www.example.com/failure",
    "customer_phone": "+1987987987987",
    "customer_email": "travis@example.com",
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>webmoney</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>100</amount>
    <currency>USD</currency>
    <return_success_url>http://www.example.com/success</return_success_url>
    <return_failure_url>http://www.example.com/failure</return_failure_url>
    <customer_phone>+1987987987987</customer_phone>
    <customer_email>travis@example.com</customer_email>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>10178</zip_code>
      <city>Los Angeles</city>
      <state>CA</state>
      <country>US</country>
    </billing_address>
  </payment_transaction>'
```

Payout Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Wallets\WebMoney');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setIsPayout('true')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setCustomerPhone('+1987987987987')
        ->setCustomerEmail('travis@example.com')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.wallets.WebMoneyRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;
import java.net.MalformedURLException;
import java.net.URL;

public class GenesisExample {
  public static void main() throws MalformedURLException {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    WebMoneyRequest request = new WebMoneyRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setIsPayout("true");
    request.setReturnSuccessUrl(new URL("http://www.example.com/success"));
    request.setReturnFailureUrl(new URL("http://www.example.com/failure"));
    request.setCustomerPhone("+1987987987987");
    request.setCustomerEmail("travis@example.com");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.webmoney(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "is_payout": "true",
    "return_success_url": "http://www.example.com/success",
    "return_failure_url": "http://www.example.com/failure",
    "customer_phone": "+1987987987987",
    "customer_email": "travis@example.com",
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>webmoney</transaction_type>
   <transaction_id>43671</transaction_id>
   <usage>40208 concert tickets</usage>
   <remote_ip>245.253.2.12</remote_ip>
   <amount>100</amount>
   <currency>USD</currency>
   <is_payout>true</is_payout>
   <return_success_url>http://www.example.com/success</return_success_url>
   <return_failure_url>http://www.example.com/failure</return_failure_url>
   <customer_phone>+1987987987987</customer_phone>
   <customer_email>travis@example.com</customer_email>
   <billing_address>
     <first_name>Travis</first_name>
     <last_name>Pastrana</last_name>
     <address1>Muster Str. 12</address1>
     <zip_code>10178</zip_code>
     <city>Los Angeles</city>
     <state>CA</state>
     <country>US</country>
   </billing_address>
 </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **webmoney** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| is_payout | required* | string | Value:true/false Flag for payout transaction |
| customer_account_id | required* | string(12) | Webmoney account ID (WMID). This field is required if is payout is set to "true" |
| customer_email | required | e-mail address | Must contain valid e-mail of customer |
| customer_phone | required* | string(32) | Must contain valid phone number of customer |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |

| Parameter | Required | Format | Description |
|---|---|---|---|
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

Auth Response

```
stdClass Object
(
    [transaction_type] => webmoney
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:17.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[webmoney]>
    <status content=[pending_async]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <redirect_url content=[https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61]>
    <timestamp content=[2020-07-15T12:08:17Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[true]>
]>
```

```
{
    transaction_type: "webmoney",
    status: "pending_async",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    redirect_url: "https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61",
    timestamp: "2020-07-15T12:08:17Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "true",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>webmoney</transaction_type>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
  <timestamp>2020-07-15T12:08:17Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

Payout Response

```
stdClass Object
(
    [transaction_type] => webmoney
    [status] => approved
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:17.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[webmoney]>
    <status content=[approved]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <timestamp content=[2020-07-15T12:08:17Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[true]>
]>
```

```
{
    transaction_type: "webmoney",
    status: "approved",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    timestamp: "2020-07-15T12:08:17Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "true",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>webmoney</transaction_type>
  <status>approved</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <timestamp>2020-07-15T12:08:17Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |

| Parameter | Type | Description |
|---|---|---|
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [transaction_type] => webmoney
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 110
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:17.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => false
)
```

```
<payment_response content=[
    <transaction_type content=[webmoney]>
    <status content=[error]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <code content=[110]>
    <message content=[Something went wrong, please contact support!]>
    <timestamp content=[2020-07-15T12:08:17Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[false]>
]>
```

```
{
    transaction_type: "webmoney",
    status: "error",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    code: "110",
    message: "Something went wrong, please contact support!",
    timestamp: "2020-07-15T12:08:17Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "false",
}
```

```
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>webmoney</transaction_type>
    <status>error</status>
    <mode>live</mode>
    <transaction_id>43671</transaction_id>
    <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
    <code>110</code>
    <message>Something went wrong, please contact support!</message>
    <timestamp>2020-07-15T12:08:17Z</timestamp>
    <descriptor>Descriptor one</descriptor>
    <amount>100</amount>
    <currency>USD</currency>
    <sent_to_acquirer>false</sent_to_acquirer>
  </payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |

| Parameter | Type | Description |
|---|---|---|
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

#### WECHAT

WeChat Pay solution offers merchants access to the over 300 million WeChat users that have linked payment accounts to their WeChat account. The solution works on desktop and mobile via a QR code generation platform.

Wechat payment transaction - after a successful validation of transaction parameters, transaction status is set to pending async and the consumer is redirected to page that contains the QR Code for scan. The consumer then opens the WeChat application on phone and scans the QR Code for payment confirmation. The gateway waits for an async notification from PaySec with the payment result of the consumer bank payment and updates the transaction status accordingly.

When the payment reaches a final state, the gateway sends a notification to the merchant on the configured notification URL for the merchant.

**Supported countries:** All countries are supported

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\OnlineBankingPayments\WeChat');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setNotificationUrl('https://www.example.com/notification')
        ->setReturnUrl('https://staging.gate.e-comprocessing.net/redirect/to_acquirer/a55ab44d242f')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setProductCode('product-code')
        ->setProductNum('1234')
        ->setProductDesc('Product description')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.oBeP.WechatRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;
import java.net.MalformedURLException;
import java.net.URL;

public class GenesisExample {
  public static void main() throws MalformedURLException {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    WechatRequest request = new WechatRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setNotificationUrl(new URL("https://www.example.com/notification"));
    request.setReturnUrl(new URL("https://staging.gate.e-comprocessing.net/redirect/to_acquirer/a55ab44d242f"));
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setProductCode("product-code");
    request.setProductNum("1234");
    request.setProductDesc("Product description");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.wechat(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "notification_url": "https://www.example.com/notification",
    "return_url": "https://staging.gate.e-comprocessing.net/redirect/to_acquirer/a55ab44d242f",
    "amount": "100",
    "currency": "USD",
    "product_code": "product-code",
    "product_num": "1234",
    "product_desc": "Product description",
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>wechat</transaction_type>
   <transaction_id>43671</transaction_id>
   <usage>40208 concert tickets</usage>
   <remote_ip>245.253.2.12</remote_ip>
   <notification_url>https://www.example.com/notification</notification_url>
   <return_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/a55ab44d242f</return_url>
   <amount>100</amount>
   <currency>USD</currency>
   <product_code>product-code</product_code>
   <product_num>1234</product_num>
   <product_desc>Product description</product_desc>
   <billing_address>
     <first_name>Travis</first_name>
     <last_name>Pastrana</last_name>
     <address1>Muster Str. 12</address1>
     <zip_code>10178</zip_code>
     <city>Los Angeles</city>
     <state>CA</state>
     <country>US</country>
   </billing_address>
 </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **wechat** |
| transaction_id | required | string(30) | Unique transaction id defined by merchant |
| usage | required | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| notification_url | required | url | URL at merchant where gateway sends outcome of transaction. |
| return_url | required | url | URL where consumer is sent to after payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details<br>Max amount in minor currency unit: 999999 |
| currency | required | string(3) | Currency code in ISO 4217 |
| product_code | optional | string(60) | Product code |
| product_num | optional | integer(10) | Product number |
| product_desc | optional | string(255) | Product description |
| customer_email | required* | e-mail address | Must contain valid e-mail of customer |
| customer_phone | required* | string(32) | Must contain valid phone number of customer |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

Successful Response

```
stdClass Object
(
    [transaction_type] => wechat
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:17.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[wechat]>
    <status content=[pending_async]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <redirect_url content=[https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61]>
    <timestamp content=[2020-07-15T12:08:17Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[true]>
]>
```

```
{
    transaction_type: "wechat",
    status: "pending_async",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    redirect_url: "https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61",
    timestamp: "2020-07-15T12:08:17Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "true",
}
```

```
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>wechat</transaction_type>
    <status>pending_async</status>
    <mode>live</mode>
    <transaction_id>43671</transaction_id>
    <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
    <technical_message>Transaction successful!</technical_message>
    <message>Transaction successful!</message>
    <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
    <timestamp>2020-07-15T12:08:17Z</timestamp>
    <descriptor>Descriptor one</descriptor>
    <amount>100</amount>
    <currency>USD</currency>
    <sent_to_acquirer>true</sent_to_acquirer>
  </payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(30) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |

| Parameter | Type | Description |
|---|---|---|
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [transaction_type] => wechat
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 110
    [message] => Please check input data for errors!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:17.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[wechat]>
    <status content=[error]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <code content=[110]>
    <message content=[Please check input data for errors!]>
    <timestamp content=[2020-07-15T12:08:17Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[true]>
]>
```

```
{
    transaction_type: "wechat",
    status: "error",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    code: "110",
    message: "Please check input data for errors!",
    timestamp: "2020-07-15T12:08:17Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "true",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>wechat</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>110</code>
  <message>Please check input data for errors!</message>
  <timestamp>2020-07-15T12:08:17Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |

| Parameter | Type | Description |
|---|---|---|
| transaction_id | string(30) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

#### ALIPAY

Alipay is an oBeP-style alternative payment method that allows you to pay directly with your ebank account.

After initiating a transaction Alipay will redirect you to their page.

There you will see a picture of a QR code, which you will have to scan with your Alipay mobile application.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\OnlineBankingPayments\Alipay');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setNotificationUrl('https://www.example.com/notification')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setAmount('100')
        ->setCurrency('CNY')
        ->setCustomerEmail('travis@example.com')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.oBeP.AlipayRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;
import java.net.MalformedURLException;
import java.net.URL;

public class GenesisExample {
  public static void main() throws MalformedURLException {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    AlipayRequest request = new AlipayRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setNotificationUrl(new URL("https://www.example.com/notification"));
    request.setReturnSuccessUrl(new URL("http://www.example.com/success"));
    request.setReturnFailureUrl(new URL("http://www.example.com/failure"));
    request.setAmount(new BigDecimal(100));
    request.setCurrency("CNY");
    request.setCustomerEmail("travis@example.com");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.alipay(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "notification_url": "https://www.example.com/notification",
    "return_success_url": "http://www.example.com/success",
    "return_failure_url": "http://www.example.com/failure",
    "amount": "100",
    "currency": "CNY",
    "customer_email": "travis@example.com",
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```xml
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>alipay</transaction_type>
   <transaction_id>43671</transaction_id>
   <usage>40208 concert tickets</usage>
   <remote_ip>245.253.2.12</remote_ip>
   <notification_url>https://www.example.com/notification</notification_url>
   <return_success_url>http://www.example.com/success</return_success_url>
   <return_failure_url>http://www.example.com/failure</return_failure_url>
   <amount>100</amount>
   <currency>CNY</currency>
   <customer_email>travis@example.com</customer_email>
   <billing_address>
     <first_name>Travis</first_name>
     <last_name>Pastrana</last_name>
     <address1>Muster Str. 12</address1>
     <zip_code>10178</zip_code>
     <city>Los Angeles</city>
     <state>CA</state>
     <country>US</country>
   </billing_address>
 </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **alipay** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| customer_email | optional | e-mail address | Must contain valid e-mail of customer |
| customer_phone | optional | string(32) | Must contain valid phone number of customer |
| birth_date | optional | dd-mm-yyyy | Required for Visa only when MCC is a Financial Services one (e.g. MCC 6012) |
| **billing_address** | optional | | See Required vs Optional API params for details |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

**Supported currencies**

| Currency name | Currency code |
|---|---|
| China yen | CNY |
| Euro | EUR |

Successful Response

```
stdClass Object
(
    [transaction_type] => alipay
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:17.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => CNY
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[alipay]>
    <status content=[pending_async]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <redirect_url content=[https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61]>
    <timestamp content=[2020-07-15T12:08:17Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[CNY]>
    <sent_to_acquirer content=[true]>
]>
```

```
{
    transaction_type: "alipay",
    status: "pending_async",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    redirect_url: "https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61",
    timestamp: "2020-07-15T12:08:17Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "CNY",
    sent_to_acquirer: "true",
}
```

```
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>alipay</transaction_type>
    <status>pending_async</status>
    <mode>live</mode>
    <transaction_id>43671</transaction_id>
    <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
    <technical_message>Transaction successful!</technical_message>
    <message>Transaction successful!</message>
    <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
    <timestamp>2020-07-15T12:08:17Z</timestamp>
    <descriptor>Descriptor one</descriptor>
    <amount>100</amount>
    <currency>CNY</currency>
    <sent_to_acquirer>true</sent_to_acquirer>
  </payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |

| Parameter | Type | Description |
|---|---|---|
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [transaction_type] => alipay
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 110
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:17.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => CNY
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[alipay]>
    <status content=[error]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <code content=[110]>
    <message content=[Something went wrong, please contact support!]>
    <timestamp content=[2020-07-15T12:08:17Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[CNY]>
    <sent_to_acquirer content=[true]>
]>
```

```
{
    transaction_type: "alipay",
    status: "error",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    code: "110",
    message: "Something went wrong, please contact support!",
    timestamp: "2020-07-15T12:08:17Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "CNY",
    sent_to_acquirer: "true",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>alipay</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>110</code>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:17Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>CNY</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |

| Parameter | Type | Description |
|---|---|---|
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

#### EZEEWALLET

eZeeWallet is a comprehensive digital wallet, connecting multiple payment methods simultaneously. It has the highest level of security, offering you a unique way to make secure and quick online payments. Consumers can upload funds to your account using P24, Trustly, Klarna (Sofort), GiroPay, iDeal, eps, NeoSurf and bank transfer and pay fast for goods and services with their available eZeeWallet balances.

> eZeeWallet and Genesis merchant accounts should be synced since authentication to eZeeWallet is done with Genesis merchant credentials

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Wallets\eZeeWallet');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setNotificationUrl('https://www.example.com/notification')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setSourceWalletId('john@example.com')
        ->setSourceWalletPwd('UDBydsDBrYWxAQA==\n')
        ->setAmount('100')
        ->setCurrency('USD');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.wallets.EzeewalletRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;
import java.net.MalformedURLException;
import java.net.URL;

public class GenesisExample {
  public static void main() throws MalformedURLException {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    EzeewalletRequest request = new EzeewalletRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setNotificationUrl(new URL("https://www.example.com/notification"));
    request.setReturnSuccessUrl(new URL("http://www.example.com/success"));
    request.setReturnFailureUrl(new URL("http://www.example.com/failure"));
    request.setSourceWalletId("john@example.com");
    request.setSourceWalletPwd("UDBydsDBrYWxAQA==\n");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.ezeewallet(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "notification_url": "https://www.example.com/notification",
    "return_success_url": "http://www.example.com/success",
    "return_failure_url": "http://www.example.com/failure",
    "source_wallet_id": "john@example.com",
    "source_wallet_pwd": "UDBydsDBrYWxAQA==\\n",
    "amount": "100",
    "currency": "USD"
  }
).send()
  .then(success)
  .catch(failure);
```

```bash
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>ezeewallet</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <notification_url>https://www.example.com/notification</notification_url>
    <return_success_url>http://www.example.com/success</return_success_url>
    <return_failure_url>http://www.example.com/failure</return_failure_url>
    <source_wallet_id>john@example.com</source_wallet_id>
    <source_wallet_pwd>UDBydsDBrYWxAQA==\n</source_wallet_pwd>
    <amount>100</amount>
    <currency>USD</currency>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **ezeewallet** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| source_wallet_id | required | string(255) | Email address of consumer who owns the wallet |
| source_wallet_pwd | required | string(255) | Password of consumer who owns the wallet, in **Base64 encoded form** |
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| notification_url | optional | url | URL at merchant where gateway sends outcome of transaction. |

`required* = conditionally required`

return_success_url, return_failure_url and notification_url are used in case of top-up, i.e the customer does not have enough money to pay for the product

Successful Response

```
stdClass Object
(
    [transaction_type] => ezeewallet
    [status] => approved
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:18.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[ezeewallet]>
    <status content=[approved]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <timestamp content=[2020-07-15T12:08:18Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[true]>
]>
```

```
{
    transaction_type: "ezeewallet",
    status: "approved",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    timestamp: "2020-07-15T12:08:18Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "true",
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>ezeewallet</transaction_type>
  <status>approved</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <timestamp>2020-07-15T12:08:18Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```


**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Pending Response


```
stdClass Object
(
    [transaction_type] => ezeewallet
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:18.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```



```
<payment_response content=[
    <transaction_type content=[ezeewallet]>
    <status content=[pending_async]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <redirect_url content=[https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61]>
    <timestamp content=[2020-07-15T12:08:18Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[true]>
]>
```

```
{
    transaction_type: "ezeewallet",
    status: "pending_async",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    redirect_url: "https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61",
    timestamp: "2020-07-15T12:08:18Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "true",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>ezeewallet</transaction_type>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
  <timestamp>2020-07-15T12:08:18Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```


**Pending Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response


```
stdClass Object
(
    [transaction_type] => ezeewallet
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 930
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:18.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => false
)
```

```
<payment_response content=[
    <transaction_type content=[ezeewallet]>
    <status content=[error]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <code content=[930]>
    <message content=[Something went wrong, please contact support!]>
    <timestamp content=[2020-07-15T12:08:18Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[false]>
]>
```

```
{
    transaction_type: "ezeewallet",
    status: "error",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    code: "930",
    message: "Something went wrong, please contact support!",
    timestamp: "2020-07-15T12:08:18Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "false",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>ezeewallet</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>930</code>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:18Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>false</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |

| Parameter | Type | Description |
|---|---|---|
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

---

# Vouchers

**CASHU**

CashU transactions are only asynchronous. After a successful validation of transaction parameters, transaction status is set to pending async, the user is redirected to CashU authentication page where he enters additional information to finish the payment. When the payment reaches a final state Genesis gateway sends notification to merchant on the configured url into its account.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Vouchers\CashU');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setAmount('100')
        ->setCurrency('EUR')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.apm.CashURequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;
import java.net.MalformedURLException;
import java.net.URL;

public class GenesisExample {
  public static void main() throws MalformedURLException {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    CashURequest request = new CashURequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setReturnSuccessUrl(new URL("http://www.example.com/success"));
    request.setReturnFailureUrl(new URL("http://www.example.com/failure"));
    request.setAmount(new BigDecimal(100));
    request.setCurrency("EUR");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.cashu(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "return_success_url": "http://www.example.com/success",
    "return_failure_url": "http://www.example.com/failure",
    "amount": "100",
    "currency": "EUR",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>cashu</transaction_type>
   <transaction_id>43671</transaction_id>
   <usage>40208 concert tickets</usage>
   <remote_ip>245.253.2.12</remote_ip>
   <return_success_url>http://www.example.com/success</return_success_url>
   <return_failure_url>http://www.example.com/failure</return_failure_url>
   <amount>100</amount>
   <currency>EUR</currency>
   <customer_email>travis@example.com</customer_email>
   <customer_phone>+1987987987987</customer_phone>
   <billing_address>
     <first_name>Travis</first_name>
     <last_name>Pastrana</last_name>
     <address1>Muster Str. 12</address1>
     <zip_code>10178</zip_code>
     <city>Los Angeles</city>
     <state>CA</state>
     <country>US</country>
   </billing_address>
 </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **cashu** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| customer_email | required | e-mail address | Must contain valid e-mail of customer |
| customer_phone | required* | string(32) | Must contain valid phone number of customer |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

**Supported countries**

| Country name | Country code |
|---|---|
| Algeria | DZ |
| Bahrain | BH |
| Egypt | EG |
| Gambia | GM |
| Ghana | GH |
| India | IN |
| Iran | IR |
| Iraq | IQ |
| Israel | IL |
| Jordan | JO |
| Kenya | KE |
| Korea | KP |
| Kuwait | KW |
| Lebanon | LB |
| Libya | LY |
| Malaysia | MY |
| Mauritania | MR |
| Morocco | MA |
| Nigeria | NG |
| Oman | OM |
| Pakistan | PK |
| Palestine | PS |
| Qatar | QA |
| Saudi Arabia | SA |
| Sierra Leone | SL |
| Sudan | SD |
| Syria | SY |
| Tanzania | TZ |
| Tunisia | TN |
| Turkey | TR |
| United Arab Emirates | AE |
| United States | US |
| Yemen | YE |

**Supported currencies**

| Currency name | Currency code |
|---|---|
| Algerian Dinar | DZD |
| AmericanDollar | USD |
| EgyptianPound | EGP |
| Euro | EUR |
| JordanianDinar | JOD |
| LebanesePound | LBP |

| Currency name | Currency code |
|---|---|
| MoroccanDirham | MAD |
| Qatar Riyal | QAR |
| Saudi Riyal | SAR |
| Turkish Lira | TRY |
| UAE Dirham | AED |

Successful Response

```
stdClass Object
(
    [transaction_type] => cashu
    [status] => approved
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:18.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => EUR
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[cashu]>
    <status content=[approved]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <redirect_url content=[https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61]>
    <timestamp content=[2020-07-15T12:08:18Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[EUR]>
    <sent_to_acquirer content=[true]>
]>
```

```
{
    transaction_type: "cashu",
    status: "approved",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    redirect_url: "https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61",
    timestamp: "2020-07-15T12:08:18Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "EUR",
    sent_to_acquirer: "true",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>cashu</transaction_type>
  <status>approved</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
  <timestamp>2020-07-15T12:08:18Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>EUR</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [transaction_type] => cashu
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 110
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:18.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => EUR
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[cashu]>
    <status content=[error]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <code content=[110]>
    <message content=[Something went wrong, please contact support!]>
    <timestamp content=[2020-07-15T12:08:18Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[EUR]>
    <sent_to_acquirer content=[true]>
]>
```

```
{
    transaction_type: "cashu",
    status: "error",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    code: "110",
    message: "Something went wrong, please contact support!",
    timestamp: "2020-07-15T12:08:18Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "EUR",
    sent_to_acquirer: "true",
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>cashu</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>110</code>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:18Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>EUR</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

**NEOSURF**

Neosurf is a prepaid card (voucher) that is used for online shopping. The card is available in over 100,000 stores worldwide, where customers can buy the prepaid vouchers, denominated up to EUR 250.00 or its equivalent in other currencies. This transaction is synchronous.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Vouchers\Neosurf');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('EUR')
        ->setVoucherNumber('')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Barney')
        ->setBillingLastName('Rubble')
        ->setBillingAddress1('23, Doestreet')
        ->setBillingZipCode('11923')
        ->setBillingCity('New York City')
        ->setBillingCountry('DE');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```curl
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>neosurf</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>100</amount>
    <currency>EUR</currency>
    <payment_transaction>voucher_number</payment_transaction>
    <customer_email>travis@example.com</customer_email>
    <customer_phone>+1987987987987</customer_phone>
    <billing_address>
      <first_name>Barney</first_name>
      <last_name>Rubble</last_name>
      <address1>23, Doestreet</address1>
      <zip_code>11923</zip_code>
      <city>New York City</city>
      <country>DE</country>
    </billing_address>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **neosurf** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| voucher_number | required | string(10) | Voucher number. Alphanumeric maximum 10 characters |

| Parameter | Required | Format | Description |
|---|---|---|---|
| customer_email | required* | e-mail address | Must contain valid e-mail of customer |
| customer_phone | required* | string(32) | Must contain valid phone number of customer |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

**Supported countries**

| Country Name | Country code |
|---|---|
| Austria | AT |
| Algeria | DZ |
| Australia | AU |
| Burundi | BI |
| Burkina Faso | BF |
| Benin | BJ |
| Belgium | BE |
| Cape Verde | CV |
| Cyprus | CY |
| Canada | CA |
| Central African Republic | CF |
| Chad | TD |
| Colombia | CO |
| Congo | CG |
| Cameroon | CM |
| Democratic Republic of Congo | CD |
| Denmark | DK |
| Equatorial Guinea | GQ |
| France | FR |
| Gambia | GM |

| Country Name | Country code |
|---|---|
| Germany | DE |
| Gabon | GA |
| Guinea | GN |
| Ghana | GH |
| Guinea-Bissau | GW |
| Hong Kong | HK |
| Ireland | IE |
| Italy | IT |
| Ivory Coast | CI |
| Kenya | KE |
| Luxembourg | LU |
| Malawi | MW |
| Mozambique | MZ |
| Morocco | MA |
| Mauritania | MR |
| Mali | ML |
| Niger | NE |
| Nigeria | NG |
| Netherlands | NL |
| New Zealand | NZ |
| Norway | NO |
| Poland | PL |
| Portugal | PT |
| Rwanda | RW |
| Russia | RU |
| Romania | RO |
| Sweden | SE |
| Spain | ES |
| Sierra Leone | SL |
| Senegal | SN |
| Sao Tome and Principe | ST |
| Switzerland | CH |
| Serbia | RS |
| Turkey | TR |
| Togo | TG |
| Tunisia | TN |
| United Kingdom | GB |
| United Republic of Tanzania | TZ |
| Uganda | UG |
| Zimbabwe | ZW |
| Zambia | ZM |

Successful Response

```
stdClass Object
(
    [transaction_type] => neosurf
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:18.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => EUR
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>neosurf</transaction_type>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <timestamp>2020-07-15T12:08:18Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>EUR</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [transaction_type] => neosurf
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 110
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:18.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => EUR
    [sent_to_acquirer] => false
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>neosurf</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>110</code>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:18Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>EUR</currency>
  <sent_to_acquirer>false</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

**PAYSAFECARD**

Paysafecard transactions are only asynchronous. After a successful validation of transaction param- eters transaction status is set to pending async the user is redirected to Paysafecard authentication page where he enters additional information to finish the payment. When the payment reaches a final state Genesis gateway sends notification to merchant on the configured url into its account.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Vouchers\Paysafecard');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.apm.PaySafeCardRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;
import java.net.MalformedURLException;
import java.net.URL;

public class GenesisExample {
  public static void main() throws MalformedURLException {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    PaySafeCardRequest request = new PaySafeCardRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setReturnSuccessUrl(new URL("http://www.example.com/success"));
    request.setReturnFailureUrl(new URL("http://www.example.com/failure"));
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.paysafecard(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "return_success_url": "http://www.example.com/success",
    "return_failure_url": "http://www.example.com/failure",
    "amount": "100",
    "currency": "USD",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>paysafecard</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <return_success_url>http://www.example.com/success</return_success_url>
    <return_failure_url>http://www.example.com/failure</return_failure_url>
    <amount>100</amount>
    <currency>USD</currency>
    <customer_email>travis@example.com</customer_email>
    <customer_phone>+1987987987987</customer_phone>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>10178</zip_code>
      <city>Los Angeles</city>
      <state>CA</state>
      <country>US</country>
    </billing_address>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **paysafecard** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| customer_email | required | e-mail address | Must contain valid e-mail of customer |
| customer_phone | required* | string(32) | Must contain valid phone number of customer |
| **billing_address** | required | | See Required vs Optional API params for details |

| Parameter | Required | Format | Description |
|---|---|---|---|
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

**Supported countries:**

| Country name | Country code |
|---|---|
| Australia | AU |
| Austria | AT |
| Belgium | BE |
| Bulgaria | BG |
| Canada | CA |
| Croatia | HR |
| Cyprus | CY |
| Czech Republic | CZ |
| Denmark | DK |
| Finland | FI |
| France | FR |
| Georgia | GE |
| Germany | DE |
| Gibraltar | GI |
| Greece | GR |
| Hungary | HU |
| Iceland | IS |
| Ireland | IE |
| Italy | IT |
| Kuwait | KW |
| Latvia | LV |
| Liechtenstein | LI |
| Lithuania | LT |

| Country name | Country code |
|---|---|
| Luxembourg | LU |
| Malta | MT |
| Mexico | MX |
| Montenegro | ME |
| Netherlands | NL |
| New Zealand | NZ |
| Norway | NO |
| Paraguay | PY |
| Peru | PE |
| Poland | PL |
| Portugal | PT |
| Romania | RO |
| Saudi Arabia | SA |
| Slovakia | SK |
| Slovenia | SI |
| Spain | ES |
| Sweden | SE |
| Switzerland | CH |
| Turkey | TR |
| United Arab Emirates | AE |
| United Kingdom | GB |
| United States of America | US |
| Uruguay | UY |

Successful Response


```
stdClass Object
(
    [transaction_type] => paysafecard
    [status] => approved
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:18.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[paysafecard]>
    <status content=[approved]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <redirect_url content=[https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61]>
    <timestamp content=[2020-07-15T12:08:18Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[true]>
]>
```

```
{
    transaction_type: "paysafecard",
    status: "approved",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    redirect_url: "https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61",
    timestamp: "2020-07-15T12:08:18Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "true",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>paysafecard</transaction_type>
  <status>approved</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
  <timestamp>2020-07-15T12:08:18Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```


**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response


```
stdClass Object
(
    [transaction_type] => paysafecard
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 110
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:18.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => false
)
```

```xml
<payment_response content=[
    <transaction_type content=[paysafecard]>
    <status content=[error]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <code content=[110]>
    <message content=[Something went wrong, please contact support!]>
    <timestamp content=[2020-07-15T12:08:18Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[false]>
]>
```

```json
{
    transaction_type: "paysafecard",
    status: "error",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    code: "110",
    message: "Something went wrong, please contact support!",
    timestamp: "2020-07-15T12:08:18Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "false",
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>paysafecard</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>110</code>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:18Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>false</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

# Online Banking ePayments (oBeP)

Online Banking ePayments refer to payment methods that are used as an alternative to credit card payments allowing consumers to pay online with their bank account.

**BANCO DO BRASIL**

Banco do Brasil offers online bank transfer payment service.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\OnlineBankingPayments\BancoDoBrasil');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setConsumerReference('barney_rubble')
        ->setNationalId('8812128812')
        ->setBirthDate('30-12-1992')
        ->setCustomerEmail('travis@example.com')


        // Billing Address
        ->setBillingFirstName('Barney')
        ->setBillingLastName('Rubble')
        ->setBillingAddress1('14, Nerazdelni str')
        ->setBillingZipCode('1407')
        ->setBillingCity('Natal')
        ->setBillingCountry('BR')


        // Risk Params
        ->setRiskUserId('123456');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```curl
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>banco_do_brasil</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <return_success_url>http://www.example.com/success</return_success_url>
    <return_failure_url>http://www.example.com/failure</return_failure_url>
    <amount>100</amount>
    <currency>USD</currency>
    <consumer_reference>barney_rubble</consumer_reference>
    <national_id>8812128812</national_id>
    <birth_date>30-12-1992</birth_date>
    <customer_email>travis@example.com</customer_email>
    <billing_address>
      <first_name>Barney</first_name>
      <last_name>Rubble</last_name>
      <address1>14, Nerazdelni str</address1>
      <zip_code>1407</zip_code>
      <city>Natal</city>
      <country>BR</country>
    </billing_address>
    <risk_params>
      <user_id>123456</user_id>
    </risk_params>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **banco_do_brasil** |

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| consumer_reference | required | string(20) | Consumer reference is a unique consumer identifier |
| national_id | required | string(20) | National ID of the consumer. See Document ID Parameter for more details. |
| birth_date | optional | string(20) | Birth date of the customer |
| customer_email | required | e-mail address | Must contain valid e-mail of customer |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

**Supported countries:**

| Country |
|---|
| BR |

Successful Response

```
stdClass Object
(
    [transaction_type] => banco_do_brasil
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:18.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```


```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>banco_do_brasil</transaction_type>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
  <timestamp>2020-07-15T12:08:18Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```


**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [transaction_type] => banco_do_brasil
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 110
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:18.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>banco_do_brasil</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>110</code>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:18Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

**BANCOMER**

Bancomer offers two options for payments in Mexico, cash payment and bank transfer.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\OnlineBankingPayments\Bancomer');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setConsumerReference('barney_rubble')
        ->setNationalId('8812128812')
        ->setBirthDate('30-12-1992')
        ->setCustomerEmail('travis@example.com')


        // Billing Address
        ->setBillingFirstName('Barney')
        ->setBillingLastName('Rubble')
        ->setBillingAddress1('14, Nerazdelni str')
        ->setBillingZipCode('1407')
        ->setBillingCity('Mexico City')
        ->setBillingCountry('MX')


        // Risk Params
        ->setRiskUserId('123456');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```curl
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>bancomer</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <return_success_url>http://www.example.com/success</return_success_url>
    <return_failure_url>http://www.example.com/failure</return_failure_url>
    <amount>100</amount>
    <currency>USD</currency>
    <consumer_reference>barney_rubble</consumer_reference>
    <national_id>8812128812</national_id>
    <birth_date>30-12-1992</birth_date>
    <customer_email>travis@example.com</customer_email>
    <billing_address>
      <first_name>Barney</first_name>
      <last_name>Rubble</last_name>
      <address1>14, Nerazdelni str</address1>
      <zip_code>1407</zip_code>
      <city>Mexico City</city>
      <country>MX</country>
    </billing_address>
    <risk_params>
      <user_id>123456</user_id>
    </risk_params>
  </payment_transaction>'
```

### Request Parameters

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **bancomer** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |

| Parameter | Required | Format | Description |
|---|---|---|---|
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| consumer_reference | required | string(20) | Consumer reference is a unique consumer identifier |
| national_id | required | string(20) | National ID of the consumer. See Document ID Parameter for more details. |
| birth_date | optional | string(20) | Birth date of the customer |
| customer_email | required | e-mail address | Must contain valid e-mail of customer |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

**Supported countries:**

| Country |
|---|
| MX |

Successful Response

```
stdClass Object
(
    [transaction_type] => bancomer
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:18.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>bancomer</transaction_type>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
  <timestamp>2020-07-15T12:08:18Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [transaction_type] => bancomer
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 110
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:18.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>bancomer</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>110</code>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:18Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

**BRADESCO**

Bradesco is a payment service in Brazil

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\OnlineBankingPayments\Bradesco');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setConsumerReference('barney_rubble')
        ->setNationalId('8812128812')
        ->setBirthDate('30-12-1992')
        ->setCustomerEmail('travis@example.com')


        // Billing Address
        ->setBillingFirstName('Barney')
        ->setBillingLastName('Rubble')
        ->setBillingAddress1('14, Nerazdelni str')
        ->setBillingZipCode('1407')
        ->setBillingCity('Rio de Janeiro')
        ->setBillingCountry('BR')


        // Risk Params
        ->setRiskUserId('123456');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```curl
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>bradesco</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <return_success_url>http://www.example.com/success</return_success_url>
    <return_failure_url>http://www.example.com/failure</return_failure_url>
    <amount>100</amount>
    <currency>USD</currency>
    <consumer_reference>barney_rubble</consumer_reference>
    <national_id>8812128812</national_id>
    <birth_date>30-12-1992</birth_date>
    <customer_email>travis@example.com</customer_email>
    <billing_address>
      <first_name>Barney</first_name>
      <last_name>Rubble</last_name>
      <address1>14, Nerazdelni str</address1>
      <zip_code>1407</zip_code>
      <city>Rio de Janeiro</city>
      <country>BR</country>
    </billing_address>
    <risk_params>
      <user_id>123456</user_id>
    </risk_params>
  </payment_transaction>'
```

### Request Parameters

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **bradesco** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |

| Parameter | Required | Format | Description |
|---|---|---|---|
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| consumer_reference | required | string(20) | Consumer reference is a unique consumer identifier |
| national_id | required | string(20) | National ID of the consumer. See Document ID Parameter for more details. |
| birth_date | optional | string(20) | Birth date of the customer |
| customer_email | required | e-mail address | Must contain valid e-mail of customer |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

**Supported countries:**

| Country |
|---|
| BR |

Successful Response

```
stdClass Object
(
    [transaction_type] => bradesco
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:18.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>bradesco</transaction_type>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
  <timestamp>2020-07-15T12:08:18Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [transaction_type] => bradesco
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 110
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:18.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>bradesco</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>110</code>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:18Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

**EPS**

EPS is the main bank transfer payment method in Austria. Every transaction is guaranteed via the scheme.

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>eps</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <return_success_url>http://www.example.com/success</return_success_url>
    <return_failure_url>http://www.example.com/failure</return_failure_url>
    <amount>100</amount>
    <currency>EUR</currency>
    <billing_address>
      <first_name>Barney</first_name>
      <last_name>Rubble</last_name>
      <address1>14, Nerazdelni str</address1>
      <zip_code>1407</zip_code>
      <city>Vienna</city>
      <country>AT</country>
    </billing_address>
  </payment_transaction>'
```

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Alternatives\PPRO');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setPaymentType('eps')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setAmount('100')
        ->setCurrency('EUR')


        // Billing Address
        ->setBillingFirstName('Barney')
        ->setBillingLastName('Rubble')
        ->setBillingAddress1('14, Nerazdelni str')
        ->setBillingZipCode('1407')
        ->setBillingCity('Vienna')
        ->setBillingCountry('AT');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.apm.PProRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;
import java.net.MalformedURLException;
import java.net.URL;

public class GenesisExample {
  public static void main() throws MalformedURLException {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    PProRequest request = new PProRequest();

    request.setTransactionId("43671");
    request.setPaymentType("eps");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setReturnSuccessUrl(new URL("http://www.example.com/success"));
    request.setReturnFailureUrl(new URL("http://www.example.com/failure"));
    request.setAmount(new BigDecimal(100));
    request.setCurrency("EUR");


    // Billing Address
    request.setBillingFirstname("Barney");
    request.setBillingLastname("Rubble");
    request.setBillingPrimaryAddress("14, Nerazdelni str");
    request.setBillingZipCode("1407");
    request.setBillingCity("Vienna");
    request.setBillingCountry("AT");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.ppro(
  {
    "transaction_id": "43671",
    "payment_type": "eps",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "return_success_url": "http://www.example.com/success",
    "return_failure_url": "http://www.example.com/failure",
    "amount": "100",
    "currency": "EUR",
    "billing_address": {
      "first_name": "Barney",
      "last_name": "Rubble",
      "address1": "14, Nerazdelni str",
      "zip_code": "1407",
      "city": "Vienna",
      "country": "AT"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>ppro</transaction_type>
   <transaction_id>43671</transaction_id>
   <payment_type>eps</payment_type>
   <usage>40208 concert tickets</usage>
   <remote_ip>245.253.2.12</remote_ip>
   <return_success_url>http://www.example.com/success</return_success_url>
   <return_failure_url>http://www.example.com/failure</return_failure_url>
   <amount>100</amount>
   <currency>EUR</currency>
   <billing_address>
     <first_name>Barney</first_name>
     <last_name>Rubble</last_name>
     <address1>14, Nerazdelni str</address1>
     <zip_code>1407</zip_code>
     <city>Vienna</city>
     <country>AT</country>
   </billing_address>
 </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | **ppro** or **eps**. Contact tech support at tech-support@e-comprocessing.com for more details. |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| payment_type | required[1] | eps | EPS. Contact tech support for more details. |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

`1 - payment_type must be submitted only when the transaction type is set to ppro`

**Supported countries and countries**

| Currency code | Country code |
|---|---|
| EUR | AT |

Successful Response

```
stdClass Object
(
    [transaction_type] => ppro
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:18.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => EUR
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[ppro]>
    <status content=[pending_async]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <redirect_url content=[https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61]>
    <timestamp content=[2020-07-15T12:08:18Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[EUR]>
    <sent_to_acquirer content=[true]>
]>
```

```
{
    transaction_type: "ppro",
    status: "pending_async",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    redirect_url: "https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61",
    timestamp: "2020-07-15T12:08:18Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "EUR",
    sent_to_acquirer: "true",
}
```

```
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>ppro</transaction_type>
    <status>pending_async</status>
    <mode>live</mode>
    <transaction_id>43671</transaction_id>
    <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
    <technical_message>Transaction successful!</technical_message>
    <message>Transaction successful!</message>
    <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
    <timestamp>2020-07-15T12:08:18Z</timestamp>
    <descriptor>Descriptor one</descriptor>
    <amount>100</amount>
    <currency>EUR</currency>
    <sent_to_acquirer>true</sent_to_acquirer>
  </payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |

| Parameter | Type | Description |
|---|---|---|
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [transaction_type] => ppro
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 110
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:18.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => EUR
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[ppro]>
    <status content=[error]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <code content=[110]>
    <message content=[Something went wrong, please contact support!]>
    <timestamp content=[2020-07-15T12:08:18Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[EUR]>
    <sent_to_acquirer content=[true]>
]>
```

```
{
    transaction_type: "ppro",
    status: "error",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    code: "110",
    message: "Something went wrong, please contact support!",
    timestamp: "2020-07-15T12:08:18Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "EUR",
    sent_to_acquirer: "true",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>ppro</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>110</code>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:18Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>EUR</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

**GIROPAY**

GiroPay is a popular real-time bank transfer payment method in Germany that integrates more than 1,500 German banks.

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>giropay</transaction_type>
  <transaction_id>43671</transaction_id>
  <usage>40208 concert tickets</usage>
  <remote_ip>245.253.2.12</remote_ip>
  <return_success_url>http://www.example.com/success</return_success_url>
  <return_failure_url>http://www.example.com/failure</return_failure_url>
  <amount>100</amount>
  <currency>EUR</currency>
  <customer_phone>+1987987987987</customer_phone>
  <billing_address>
    <first_name>Barney</first_name>
    <last_name>Rubble</last_name>
    <address1>14, Nerazdelni str</address1>
    <zip_code>1407</zip_code>
    <city>Berlin</city>
    <country>DE</country>
  </billing_address>
</payment_transaction>'
```

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Alternatives\PPRO');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setPaymentType('giropay')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setAmount('100')
        ->setCurrency('EUR')
        ->setBic('GENODETT488')
        ->setIban('DE07444488880123456789')


        // Billing Address
        ->setBillingFirstName('Barney')
        ->setBillingLastName('Rubble')
        ->setBillingAddress1('14, Nerazdelni str')
        ->setBillingZipCode('1407')
        ->setBillingCity('Berlin')
        ->setBillingCountry('DE');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.apm.PProRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;
import java.net.MalformedURLException;
import java.net.URL;

public class GenesisExample {
  public static void main() throws MalformedURLException {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    PProRequest request = new PProRequest();

    request.setTransactionId("43671");
    request.setPaymentType("giropay");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setReturnSuccessUrl(new URL("http://www.example.com/success"));
    request.setReturnFailureUrl(new URL("http://www.example.com/failure"));
    request.setAmount(new BigDecimal(100));
    request.setCurrency("EUR");
    request.setBic("GENODETT488");
    request.setIban("DE07444488880123456789");


    // Billing Address
    request.setBillingFirstname("Barney");
    request.setBillingLastname("Rubble");
    request.setBillingPrimaryAddress("14, Nerazdelni str");
    request.setBillingZipCode("1407");
    request.setBillingCity("Berlin");
    request.setBillingCountry("DE");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.ppro(
  {
    "transaction_id": "43671",
    "payment_type": "giropay",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "return_success_url": "http://www.example.com/success",
    "return_failure_url": "http://www.example.com/failure",
    "amount": "100",
    "currency": "EUR",
    "bic": "GENODETT488",
    "iban": "DE07444488880123456789",
    "billing_address": {
      "first_name": "Barney",
      "last_name": "Rubble",
      "address1": "14, Nerazdelni str",
      "zip_code": "1407",
      "city": "Berlin",
      "country": "DE"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>ppro</transaction_type>
    <transaction_id>43671</transaction_id>
    <payment_type>giropay</payment_type>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <return_success_url>http://www.example.com/success</return_success_url>
    <return_failure_url>http://www.example.com/failure</return_failure_url>
    <amount>100</amount>
    <currency>EUR</currency>
    <bic>GENODETT488</bic>
    <iban>DE07444488880123456789</iban>
    <billing_address>
      <first_name>Barney</first_name>
      <last_name>Rubble</last_name>
      <address1>14, Nerazdelni str</address1>
      <zip_code>1407</zip_code>
      <city>Berlin</city>
      <country>DE</country>
    </billing_address>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | **ppro** or **giropay**. Contact tech support at tech-support@e-comprocessing.com for more details. |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| payment_type | required[1] | giropay | GiroPay. Contact tech support for more details. |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| bic | required | string(11) | Valid BIC string. Must be 8 or 11 characters long |
| iban | required* | string(22) | String must start with ”DE” followed by 20 digits |

| Parameter | Required | Format | Description |
|---|---|---|---|
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required | string(255) | Customer first name |
| last_name | required | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

`1 - payment_type must be submitted only when the transaction type is set to ppro`

**Supported currencies and countries**

| Currency code | Country code |
|---|---|
| EUR | DE |

Successful Response

```
stdClass Object
(
    [transaction_type] => ppro
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:18.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => EUR
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[ppro]>
    <status content=[pending_async]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <redirect_url content=[https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61]>
    <timestamp content=[2020-07-15T12:08:18Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[EUR]>
    <sent_to_acquirer content=[true]>
]>
```

```
{
    transaction_type: "ppro",
    status: "pending_async",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    redirect_url: "https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61",
    timestamp: "2020-07-15T12:08:18Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "EUR",
    sent_to_acquirer: "true",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>ppro</transaction_type>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
  <timestamp>2020-07-15T12:08:18Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>EUR</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```


**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response


```
stdClass Object
(
    [transaction_type] => ppro
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 110
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:18.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => EUR
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[ppro]>
    <status content=[error]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <code content=[110]>
    <message content=[Something went wrong, please contact support!]>
    <timestamp content=[2020-07-15T12:08:18Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[EUR]>
    <sent_to_acquirer content=[true]>
]>
```

```
{
    transaction_type: "ppro",
    status: "error",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    code: "110",
    message: "Something went wrong, please contact support!",
    timestamp: "2020-07-15T12:08:18Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "EUR",
    sent_to_acquirer: "true",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>ppro</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>110</code>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:18Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>EUR</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

**IDEBIT**

iDebit connects consumers to their online banking directly from checkout, enabling secure, real-time payments without a credit card. Using iDebit allows consumers to transfer funds to merchants without revealing their personal banking information.

The main difference between iDebit and InstaDebit is that InstaDebit uses eCheck, iDebit uses online bank transfer.

iDebit transactions have payins and payouts. The payin is asynchronous, while the payout is synchronous.

iDebit Payin transaction - after a successful validation of transaction parameters, transaction status is set to pending async and the consumer is redirected to the iDebit consumers page.

The consumer then carries out the specified transaction details and finalizes the transaction. The gateway waits for an async notification from iDebit with the payment result of the consumer bank payment executed on the iDebit consumers page, and updates the transaction status accordingly.

When the payment reaches a final state, the gateway sends a notification to the merchant on the configured notification URL for the merchant. iDebit Payout transaction - the transaction is synchronous and transaction status is set immediately after the response.

**Supported countries**

| Country name | Country code |
|---|---|
| Canada | CA |

iDebit is available only for Canadian merchants and consumers.

**PAYIN**

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\OnlineBankingPayments\iDebit\Payin');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setCustomerAccountId('1534537')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setNotificationUrl('https://www.example.com/notification')
        ->setReturnUrl('https://staging.gate.e-comprocessing.net/redirect/to_acquirer/a55ab44d242f')
        ->setAmount('100')
        ->setCurrency('CAD')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('46 Donora Dr')
        ->setBillingZipCode('M4B1B3')
        ->setBillingCity('Toronto')
        ->setBillingState('ON')
        ->setBillingCountry('CA');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.oBeP.IDebitPayInRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;
import java.net.MalformedURLException;
import java.net.URL;

public class GenesisExample {
  public static void main() throws MalformedURLException {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    IDebitPayInRequest request = new IDebitPayInRequest();

    request.setTransactionId("43671");
    request.setCustomerAccountId("1534537");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setNotificationUrl(new URL("https://www.example.com/notification"));
    request.setReturnUrl(new URL("https://staging.gate.e-comprocessing.net/redirect/to_acquirer/a55ab44d242f"));
    request.setAmount(new BigDecimal(100));
    request.setCurrency("CAD");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("46 Donora Dr");
    request.setBillingZipCode("M4B1B3");
    request.setBillingCity("Toronto");
    request.setBillingState("ON");
    request.setBillingCountry("CA");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.idebit_payin(
  {
    "transaction_id": "43671",
    "customer_account_id": "1534537",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "notification_url": "https://www.example.com/notification",
    "return_url": "https://staging.gate.e-comprocessing.net/redirect/to_acquirer/a55ab44d242f",
    "amount": "100",
    "currency": "CAD",
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "46 Donora Dr",
      "zip_code": "M4B1B3",
      "city": "Toronto",
      "state": "ON",
      "country": "CA"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>idebit_payin</transaction_type>
   <transaction_id>43671</transaction_id>
   <customer_account_id>1534537</customer_account_id>
   <usage>40208 concert tickets</usage>
   <remote_ip>245.253.2.12</remote_ip>
   <notification_url>https://www.example.com/notification</notification_url>
   <return_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/a55ab44d242f</return_url>
   <amount>100</amount>
   <currency>CAD</currency>
   <billing_address>
     <first_name>Travis</first_name>
     <last_name>Pastrana</last_name>
     <address1>46 Donora Dr</address1>
     <zip_code>M4B1B3</zip_code>
     <city>Toronto</city>
     <state>ON</state>
     <country>CA</country>
   </billing_address>
 </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **idebit_payin** |
| transaction_id | required | string(30) | Unique transaction id defined by merchant |
| usage | required* | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| customer_account_id | required | string(20) | Unique consumer account ID |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| customer_email | required* | e-mail address | Must contain valid e-mail of customer |
| customer_phone | required* | string(32) | Must contain valid phone number of customer |
| birth_date | required* | dd-mm-yyyy | Required for Visa only when MCC is a Financial Services one (e.g. MCC 6012) |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

Successful Response

```
stdClass Object
(
    [transaction_type] => idebit_payin
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:18.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => CAD
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[idebit_payin]>
    <status content=[pending_async]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <redirect_url content=[https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61]>
    <timestamp content=[2020-07-15T12:08:18Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[CAD]>
    <sent_to_acquirer content=[true]>
]>
```

```
{
    transaction_type: "idebit_payin",
    status: "pending_async",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    redirect_url: "https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61",
    timestamp: "2020-07-15T12:08:18Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "CAD",
    sent_to_acquirer: "true",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>idebit_payin</transaction_type>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
  <timestamp>2020-07-15T12:08:18Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>CAD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(30) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |

| Parameter | Type | Description |
|---|---|---|
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [transaction_type] => idebit_payin
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 320
    [message] => amount is missing!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:18.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => CAD
    [sent_to_acquirer] => false
)
```

```
<payment_response content=[
    <transaction_type content=[idebit_payin]>
    <status content=[error]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <code content=[320]>
    <message content=[amount is missing!]>
    <timestamp content=[2020-07-15T12:08:18Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[CAD]>
    <sent_to_acquirer content=[false]>
]>
```

```
{
    transaction_type: "idebit_payin",
    status: "error",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    code: "320",
    message: "amount is missing!",
    timestamp: "2020-07-15T12:08:18Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "CAD",
    sent_to_acquirer: "false",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>idebit_payin</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>320</code>
  <message>amount is missing!</message>
  <timestamp>2020-07-15T12:08:18Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>CAD</currency>
  <sent_to_acquirer>false</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |

| Parameter | Type | Description |
|---|---|---|
| transaction_id | string(30) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

**PAYOUT**

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\OnlineBankingPayments\iDebit\Payout');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setReferenceId('43672')
        ->setAmount('100')
        ->setCurrency('CAD');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.oBeP.IDebitPayOutRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    IDebitPayOutRequest request = new IDebitPayOutRequest();

    request.setTransactionId("43671");
    request.setReferenceId("43672");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("CAD");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.idebit_payout(
  {
    "transaction_id": "43671",
    "reference_id": "43672",
    "amount": "100",
    "currency": "CAD"
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>idebit_payout</transaction_type>
    <transaction_id>43671</transaction_id>
    <reference_id>43672</reference_id>
    <amount>100</amount>
    <currency>CAD</currency>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **idebit_payout** |
| transaction_id | required | string(30) | Unique transaction id defined by merchant |
| reference_id | required | string(32) | Unique id returned by corresponding transaction |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |

`required* = conditionally required`

Successful Response

```
stdClass Object
(
    [transaction_type] => idebit_payout
    [status] => approved
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:18.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [amount] => 100
    [currency] => CAD
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[idebit_payout]>
    <status content=[approved]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <timestamp content=[2020-07-15T12:08:18Z]>
    <amount content=[100]>
    <currency content=[CAD]>
    <sent_to_acquirer content=[true]>
]>
```

```
{
    transaction_type: "idebit_payout",
    status: "approved",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    timestamp: "2020-07-15T12:08:18Z",
    amount: "100",
    currency: "CAD",
    sent_to_acquirer: "true",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>idebit_payout</transaction_type>
  <status>approved</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <timestamp>2020-07-15T12:08:18Z</timestamp>
  <amount>100</amount>
  <currency>CAD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```


**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(30) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response


```
stdClass Object
(
    [transaction_type] => idebit_payout
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 110
    [technical_message] => amount is missing!
    [message] => Please check input data for errors!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:18.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [amount] => 100
    [currency] => CAD
    [sent_to_acquirer] => false
)
```

```
<payment_response content=[
    <transaction_type content=[idebit_payout]>
    <status content=[error]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <code content=[110]>
    <technical_message content=[amount is missing!]>
    <message content=[Please check input data for errors!]>
    <timestamp content=[2020-07-15T12:08:18Z]>
    <amount content=[100]>
    <currency content=[CAD]>
    <sent_to_acquirer content=[false]>
]>
```

```
{
    transaction_type: "idebit_payout",
    status: "error",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    code: "110",
    technical_message: "amount is missing!",
    message: "Please check input data for errors!",
    timestamp: "2020-07-15T12:08:18Z",
    amount: "100",
    currency: "CAD",
    sent_to_acquirer: "false",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>idebit_payout</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>110</code>
  <technical_message>amount is missing!</technical_message>
  <message>Please check input data for errors!</message>
  <timestamp>2020-07-15T12:08:18Z</timestamp>
  <amount>100</amount>
  <currency>CAD</currency>
  <sent_to_acquirer>false</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(30) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

**IDEAL**

iDeal is the most popular payment method in the Netherlands and is a real-time bank transfer system covering all major Dutch consumer banks.

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>ideal</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <return_success_url>http://www.example.com/success</return_success_url>
    <return_failure_url>http://www.example.com/failure</return_failure_url>
    <amount>100</amount>
    <currency>EUR</currency>
    <customer_phone>+1987987987987</customer_phone>
    <billing_address>
      <first_name>Barney</first_name>
      <last_name>Rubble</last_name>
      <address1>14, Nerazdelni str</address1>
      <zip_code>1407</zip_code>
      <city>Amsterdam</city>
      <country>NL</country>
    </billing_address>
  </payment_transaction>'
```

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Alternatives\PPRO');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setPaymentType('ideal')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setAmount('100')
        ->setCurrency('EUR')


        // Billing Address
        ->setBillingFirstName('Barney')
        ->setBillingLastName('Rubble')
        ->setBillingAddress1('14, Nerazdelni str')
        ->setBillingZipCode('1407')
        ->setBillingCity('Amsterdam')
        ->setBillingCountry('NL');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.apm.PProRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;
import java.net.MalformedURLException;
import java.net.URL;

public class GenesisExample {
  public static void main() throws MalformedURLException {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    PProRequest request = new PProRequest();

    request.setTransactionId("43671");
    request.setPaymentType("ideal");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setReturnSuccessUrl(new URL("http://www.example.com/success"));
    request.setReturnFailureUrl(new URL("http://www.example.com/failure"));
    request.setAmount(new BigDecimal(100));
    request.setCurrency("EUR");


    // Billing Address
    request.setBillingFirstname("Barney");
    request.setBillingLastname("Rubble");
    request.setBillingPrimaryAddress("14, Nerazdelni str");
    request.setBillingZipCode("1407");
    request.setBillingCity("Amsterdam");
    request.setBillingCountry("NL");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.ppro(
  {
    "transaction_id": "43671",
    "payment_type": "ideal",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "return_success_url": "http://www.example.com/success",
    "return_failure_url": "http://www.example.com/failure",
    "amount": "100",
    "currency": "EUR",
    "billing_address": {
      "first_name": "Barney",
      "last_name": "Rubble",
      "address1": "14, Nerazdelni str",
      "zip_code": "1407",
      "city": "Amsterdam",
      "country": "NL"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```xml
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>ppro</transaction_type>
   <transaction_id>43671</transaction_id>
   <payment_type>ideal</payment_type>
   <usage>40208 concert tickets</usage>
   <remote_ip>245.253.2.12</remote_ip>
   <return_success_url>http://www.example.com/success</return_success_url>
   <return_failure_url>http://www.example.com/failure</return_failure_url>
   <amount>100</amount>
   <currency>EUR</currency>
   <billing_address>
     <first_name>Barney</first_name>
     <last_name>Rubble</last_name>
     <address1>14, Nerazdelni str</address1>
     <zip_code>1407</zip_code>
     <city>Amsterdam</city>
     <country>NL</country>
   </billing_address>
 </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | **ppro** or **ideal**. Contact tech support at tech-support@e-comprocessing.com for more details. |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| payment_type | required[1] | ideal | iDeal. Contact tech support at tech-support@e-comprocessing.com for more details. |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| bic | optional | string(11) | SWIFT/BIC code of the customer's bank |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

```
1 - payment_type must be submitted only when the transaction type is set to ppro
```

**Supported currencies and countries**

| Currency code | Country code |
|---|---|
| EUR | NL |

Successful Response

```
stdClass Object
(
    [transaction_type] => ppro
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:19.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => EUR
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[ppro]>
    <status content=[pending_async]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <redirect_url content=[https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61]>
    <timestamp content=[2020-07-15T12:08:19Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[EUR]>
    <sent_to_acquirer content=[true]>
]>
```

```
{
    transaction_type: "ppro",
    status: "pending_async",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    redirect_url: "https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61",
    timestamp: "2020-07-15T12:08:19Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "EUR",
    sent_to_acquirer: "true",
}
```

```
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>ppro</transaction_type>
    <status>pending_async</status>
    <mode>live</mode>
    <transaction_id>43671</transaction_id>
    <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
    <technical_message>Transaction successful!</technical_message>
    <message>Transaction successful!</message>
    <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
    <timestamp>2020-07-15T12:08:19Z</timestamp>
    <descriptor>Descriptor one</descriptor>
    <amount>100</amount>
    <currency>EUR</currency>
    <sent_to_acquirer>true</sent_to_acquirer>
  </payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |

| Parameter | Type | Description |
|---|---|---|
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [transaction_type] => ppro
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 110
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:19.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => EUR
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[ppro]>
    <status content=[error]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <code content=[110]>
    <message content=[Something went wrong, please contact support!]>
    <timestamp content=[2020-07-15T12:08:19Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[EUR]>
    <sent_to_acquirer content=[true]>
]>
```

```
{
    transaction_type: "ppro",
    status: "error",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    code: "110",
    message: "Something went wrong, please contact support!",
    timestamp: "2020-07-15T12:08:19Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "EUR",
    sent_to_acquirer: "true",
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>ppro</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>110</code>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:19Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>EUR</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

#### INSTADEBIT

InstaDebit connects consumers to their online banking directly from checkout, enabling secure, realtime payments without a credit card. Using InstaDebit allows consumers to transfer funds to merchants without revealing their personal banking information.

The main difference between iDebit and InstaDebit is that InstaDebit uses eCheck, iDebit uses online bank transfer.

InstaDebit transactions have payins and payouts. The payin is asynchronous, while the payout is synchronous.

InstaDebit Payin transaction - after a successful validation of transaction parameters, transaction status is set to pending async and the consumer is redirected to the InstaDebit consumers page. The consumer then carries out the specified transaction details and finalizes the transaction. The gateway waits for an async notification from InstaDebit with the payment result of the consumer bank payment executed on the InstaDebit consumers page, and updates the transaction status accordingly.

When the payment reaches a final state, the gateway sends a notification to the merchant on the configured notification URL for the merchant.

InstaDebit Payout transaction - the transaction is synchronous and transaction status is set immediately after the response.

**Supported countries**

| Country name | Country code |
|---|---|
| Canada | CA |

InstaDebit is available only for Canadian merchants and consumers.

#### PAYIN

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\OnlineBankingPayments\InstaDebit\Payin');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setCustomerAccountId('118221674199')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setNotificationUrl('https://www.example.com/notification')
        ->setReturnUrl('https://staging.gate.e-comprocessing.net/redirect/to_acquirer/a55ab44d242f')
        ->setAmount('100')
        ->setCurrency('CAD')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('46 Donora Dr')
        ->setBillingZipCode('M4B1B3')
        ->setBillingCity('Toronto')
        ->setBillingState('ON')
        ->setBillingCountry('CA');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.oBeP.InstaDebitPayInRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;
import java.net.MalformedURLException;
import java.net.URL;

public class GenesisExample {
  public static void main() throws MalformedURLException {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    InstaDebitPayInRequest request = new InstaDebitPayInRequest();

    request.setTransactionId("43671");
    request.setCustomerAccountId("118221674199");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setNotificationUrl(new URL("https://www.example.com/notification"));
    request.setReturnUrl(new URL("https://staging.gate.e-comprocessing.net/redirect/to_acquirer/a55ab44d242f"));
    request.setAmount(new BigDecimal(100));
    request.setCurrency("CAD");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("46 Donora Dr");
    request.setBillingZipCode("M4B1B3");
    request.setBillingCity("Toronto");
    request.setBillingState("ON");
    request.setBillingCountry("CA");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.insta_debit_payin(
  {
    "transaction_id": "43671",
    "customer_account_id": "118221674199",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "notification_url": "https://www.example.com/notification",
    "return_url": "https://staging.gate.e-comprocessing.net/redirect/to_acquirer/a55ab44d242f",
    "amount": "100",
    "currency": "CAD",
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "46 Donora Dr",
      "zip_code": "M4B1B3",
      "city": "Toronto",
      "state": "ON",
      "country": "CA"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>insta_debit_payin</transaction_type>
    <transaction_id>43671</transaction_id>
    <customer_account_id>118221674199</customer_account_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <notification_url>https://www.example.com/notification</notification_url>
    <return_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/a55ab44d242f</return_url>
    <amount>100</amount>
    <currency>CAD</currency>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>46 Donora Dr</address1>
      <zip_code>M4B1B3</zip_code>
      <city>Toronto</city>
      <state>ON</state>
      <country>CA</country>
    </billing_address>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **insta_debit_payin** |
| transaction_id | required | string(30) | Unique transaction id defined by merchant |
| customer_account_id | required | string(20) | Unique consumer account ID |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| return_url | required | url | URL where consumer is sent to after payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| customer_email | required* | e-mail address | Must contain valid e-mail of customer |
| customer_phone | required* | string(32) | Must contain valid phone number of customer |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |

| Parameter | Required | Format | Description |
|---|---|---|---|
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

Successful Response

```
stdClass Object
(
    [transaction_type] => insta_debit_payin
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:19.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => CAD
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[insta_debit_payin]>
    <status content=[pending_async]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <redirect_url content=[https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61]>
    <timestamp content=[2020-07-15T12:08:19Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[CAD]>
    <sent_to_acquirer content=[true]>
]>
```

```
{
    transaction_type: "insta_debit_payin",
    status: "pending_async",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    redirect_url: "https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61",
    timestamp: "2020-07-15T12:08:19Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "CAD",
    sent_to_acquirer: "true",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>insta_debit_payin</transaction_type>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
  <timestamp>2020-07-15T12:08:19Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>CAD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(30) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [transaction_type] => insta_debit_payin
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [message] => Please check input data for errors!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:19.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => CAD
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[insta_debit_payin]>
    <status content=[error]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <message content=[Please check input data for errors!]>
    <timestamp content=[2020-07-15T12:08:19Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[CAD]>
    <sent_to_acquirer content=[true]>
]>
```

```
{
    transaction_type: "insta_debit_payin",
    status: "error",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    message: "Please check input data for errors!",
    timestamp: "2020-07-15T12:08:19Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "CAD",
    sent_to_acquirer: "true",
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>insta_debit_payin</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <message>Please check input data for errors!</message>
  <timestamp>2020-07-15T12:08:19Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>CAD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(30) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

**PAYOUT**

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\OnlineBankingPayments\InstaDebit\Payout');
    $request = $genesis->request();

    $request
        ->setReferenceId('43672')
        ->setAmount('100')
        ->setCurrency('CAD');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.oBeP.InstaDebitPayOutRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    InstaDebitPayOutRequest request = new InstaDebitPayOutRequest();

    request.setReferenceId("43672");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("CAD");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.insta_debit_payout(
  {
    "reference_id": "43672",
    "amount": "100",
    "currency": "CAD"
  }
).send()
  .then(success)
  .catch(failure);
```

```bash
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <reference_id>43672</reference_id>
    <amount>100</amount>
    <currency>CAD</currency>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **insta_debit_payout** |
| transaction_id | required | string(30) | Unique transaction id defined by merchant |
| reference_id | required | string(32) | unique id of approved InstaDebit Payin transaction. See InstaDebit Payin Response, unique id |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details<br>Max amount in minor currency unit: 999999 |
| currency | required | string(3) | Currency code in ISO 4217 |

`required* = conditionally required`

Successful Response

```
stdClass Object
(
    [transaction_type] => insta_debit_payout
    [status] => approved
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:19.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [amount] => 100
    [currency] => CAD
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[insta_debit_payout]>
    <status content=[approved]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <timestamp content=[2020-07-15T12:08:19Z]>
    <amount content=[100]>
    <currency content=[CAD]>
    <sent_to_acquirer content=[true]>
]>
```

```
{
    transaction_type: "insta_debit_payout",
    status: "approved",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    timestamp: "2020-07-15T12:08:19Z",
    amount: "100",
    currency: "CAD",
    sent_to_acquirer: "true",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>insta_debit_payout</transaction_type>
  <status>approved</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <timestamp>2020-07-15T12:08:19Z</timestamp>
  <amount>100</amount>
  <currency>CAD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(30) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [transaction_type] => insta_debit_payout
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 320
    [technical_message] => amount is missing!
    [message] => Please check input data for errors!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:19.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [amount] => 100
    [currency] => CAD
    [sent_to_acquirer] => false
)
```

```
<payment_response content=[
    <transaction_type content=[insta_debit_payout]>
    <status content=[error]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <code content=[320]>
    <technical_message content=[amount is missing!]>
    <message content=[Please check input data for errors!]>
    <timestamp content=[2020-07-15T12:08:19Z]>
    <amount content=[100]>
    <currency content=[CAD]>
    <sent_to_acquirer content=[false]>
]>
```

```
{
    transaction_type: "insta_debit_payout",
    status: "error",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    code: "320",
    technical_message: "amount is missing!",
    message: "Please check input data for errors!",
    timestamp: "2020-07-15T12:08:19Z",
    amount: "100",
    currency: "CAD",
    sent_to_acquirer: "false",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>insta_debit_payout</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>320</code>
  <technical_message>amount is missing!</technical_message>
  <message>Please check input data for errors!</message>
  <timestamp>2020-07-15T12:08:19Z</timestamp>
  <amount>100</amount>
  <currency>CAD</currency>
  <sent_to_acquirer>false</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(30) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |

| Parameter | Type | Description |
|---|---|---|
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

**INSTANTTRANSFER**

> InstantTransfer is a payment method in Germany

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\OnlineBankingPayments\InstantTransfer');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setAmount('100')
        ->setCurrency('EUR')
        ->setIban('DE07444488880123456789')
        ->setBic('GENODETT488')
        ->setCustomerEmail('travis@example.com')


        // Billing Address
        ->setBillingFirstName('Barney')
        ->setBillingLastName('Rubble')
        ->setBillingAddress1('14, Nerazdelni str')
        ->setBillingZipCode('1407')
        ->setBillingCity('Berlin')
        ->setBillingCountry('DE')


        // Risk Params
        ->setRiskUserId('123456');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>instant_transfer</transaction_type>
   <transaction_id>43671</transaction_id>
   <usage>40208 concert tickets</usage>
   <remote_ip>245.253.2.12</remote_ip>
   <return_success_url>http://www.example.com/success</return_success_url>
   <return_failure_url>http://www.example.com/failure</return_failure_url>
   <amount>100</amount>
   <currency>EUR</currency>
   <iban>DE07444488880123456789</iban>
   <bic>GENODETT488</bic>
   <customer_email>travis@example.com</customer_email>
   <billing_address>
     <first_name>Barney</first_name>
     <last_name>Rubble</last_name>
     <address1>14, Nerazdelni str</address1>
     <zip_code>1407</zip_code>
     <city>Berlin</city>
     <country>DE</country>
   </billing_address>
   <risk_params>
     <user_id>123456</user_id>
   </risk_params>
 </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **instant_transfer** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| iban | optional | string(34) | Customer's IBAN number |
| bic | optional | string(11) | SWIFT/BIC code of the customer's bank |
| customer_email | optional | e-mail address | Must contain valid e-mail of customer |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |

| Parameter | Required | Format | Description |
|---|---|---|---|
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

**Supported countries:**

| Country code |
|---|
| DE |

Successful Response

```
stdClass Object
(
    [transaction_type] => instant_transfer
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:19.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => EUR
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```


```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>instant_transfer</transaction_type>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
  <timestamp>2020-07-15T12:08:19Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>EUR</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```


**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |

| Parameter | Type | Description |
|---|---|---|
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [transaction_type] => instant_transfer
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 110
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:19.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => EUR
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>instant_transfer</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>110</code>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:19Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>EUR</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

**ITAU**

Itau is a real-time online bank transfer method and a virtual card.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\OnlineBankingPayments\Itau');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setConsumerReference('barney_rubble')
        ->setNationalId('8812128812')
        ->setBirthDate('30-12-1992')
        ->setCustomerEmail('travis@example.com')


        // Billing Address
        ->setBillingFirstName('Barney')
        ->setBillingLastName('Rubble')
        ->setBillingAddress1('14, Nerazdelni str')
        ->setBillingZipCode('1407')
        ->setBillingCity('Rio de Janeiro')
        ->setBillingCountry('BR')


        // Risk Params
        ->setRiskUserId('123456');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```curl
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>itau</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <return_success_url>http://www.example.com/success</return_success_url>
    <return_failure_url>http://www.example.com/failure</return_failure_url>
    <amount>100</amount>
    <currency>USD</currency>
    <consumer_reference>barney_rubble</consumer_reference>
    <national_id>8812128812</national_id>
    <birth_date>30-12-1992</birth_date>
    <customer_email>travis@example.com</customer_email>
    <billing_address>
      <first_name>Barney</first_name>
      <last_name>Rubble</last_name>
      <address1>14, Nerazdelni str</address1>
      <zip_code>1407</zip_code>
      <city>Rio de Janeiro</city>
      <country>BR</country>
    </billing_address>
    <risk_params>
      <user_id>123456</user_id>
    </risk_params>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **itau** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |

| Parameter | Required | Format | Description |
|---|---|---|---|
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| consumer_reference | required | string(20) | Consumer reference is a unique consumer identifier |
| national_id | required | string(20) | National ID of the consumer. See Document ID Parameter for more details. |
| birth_date | optional | string(20) | Birth date of the customer |
| customer_email | required | e-mail address | Must contain valid e-mail of customer |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

**Supported countries:**

| Country |
|---|
| BR |

Successful Response

```
stdClass Object
(
    [transaction_type] => itau
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:19.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>itau</transaction_type>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
  <timestamp>2020-07-15T12:08:19Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [transaction_type] => itau
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 110
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:19.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>itau</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>110</code>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:19Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

**MULTIBANCO**

Multibanco allows Portuguese shoppers to do payments through the Internet by using virtual credit cards

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\OnlineBankingPayments\Multibanco');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setAmount('100')
        ->setCurrency('EUR')
        ->setCustomerEmail('travis@example.com')


        // Billing Address
        ->setBillingFirstName('Barney')
        ->setBillingLastName('Rubble')
        ->setBillingAddress1('14, Nerazdelni str')
        ->setBillingZipCode('1407')
        ->setBillingCity('Porto')
        ->setBillingCountry('PT')


        // Risk Params
        ->setRiskUserId('123456');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>multibanco</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <return_success_url>http://www.example.com/success</return_success_url>
    <return_failure_url>http://www.example.com/failure</return_failure_url>
    <amount>100</amount>
    <currency>EUR</currency>
    <customer_email>travis@example.com</customer_email>
    <billing_address>
      <first_name>Barney</first_name>
      <last_name>Rubble</last_name>
      <address1>14, Nerazdelni str</address1>
      <zip_code>1407</zip_code>
      <city>Porto</city>
      <country>PT</country>
    </billing_address>
    <risk_params>
      <user_id>123456</user_id>
    </risk_params>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **multibanco** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |

| Parameter | Required | Format | Description |
|---|---|---|---|
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| customer_email | optional | e-mail address | Must contain valid e-mail of customer |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

**Supported countries:**

| Country code |
|---|
| PT |

Successful Response

```
stdClass Object
(
    [transaction_type] => multibanco
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:19.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => EUR
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>multibanco</transaction_type>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
  <timestamp>2020-07-15T12:08:19Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>EUR</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```


**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response


```
stdClass Object
(
    [transaction_type] => multibanco
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 110
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:19.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => EUR
    [sent_to_acquirer] => true
)
```


```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>multibanco</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>110</code>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:19Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>EUR</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

#### MYBANK

MyBank is an overlay banking system for Italy and Spain.

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>my_bank</transaction_type>
  <transaction_id>43671</transaction_id>
  <usage>40208 concert tickets</usage>
  <remote_ip>245.253.2.12</remote_ip>
  <return_success_url>http://www.example.com/success</return_success_url>
  <return_failure_url>http://www.example.com/failure</return_failure_url>
  <amount>100</amount>
  <currency>EUR</currency>
  <customer_phone>+1987987987987</customer_phone>
  <billing_address>
    <first_name>Barney</first_name>
    <last_name>Rubble</last_name>
    <address1>14, Nerazdelni str</address1>
    <zip_code>1407</zip_code>
    <city>Rome</city>
    <country>IT</country>
  </billing_address>
</payment_transaction>'
```

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Alternatives\PPRO');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setPaymentType('mybank')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setAmount('100')
        ->setCurrency('EUR')


        // Billing Address
        ->setBillingFirstName('Barney')
        ->setBillingLastName('Rubble')
        ->setBillingAddress1('14, Nerazdelni str')
        ->setBillingZipCode('1407')
        ->setBillingCity('Rome')
        ->setBillingCountry('IT');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.apm.PProRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;
import java.net.MalformedURLException;
import java.net.URL;

public class GenesisExample {
  public static void main() throws MalformedURLException {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    PProRequest request = new PProRequest();

    request.setTransactionId("43671");
    request.setPaymentType("mybank");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setReturnSuccessUrl(new URL("http://www.example.com/success"));
    request.setReturnFailureUrl(new URL("http://www.example.com/failure"));
    request.setAmount(new BigDecimal(100));
    request.setCurrency("EUR");


    // Billing Address
    request.setBillingFirstname("Barney");
    request.setBillingLastname("Rubble");
    request.setBillingPrimaryAddress("14, Nerazdelni str");
    request.setBillingZipCode("1407");
    request.setBillingCity("Rome");
    request.setBillingCountry("IT");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.ppro(
  {
    "transaction_id": "43671",
    "payment_type": "mybank",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "return_success_url": "http://www.example.com/success",
    "return_failure_url": "http://www.example.com/failure",
    "amount": "100",
    "currency": "EUR",
    "billing_address": {
      "first_name": "Barney",
      "last_name": "Rubble",
      "address1": "14, Nerazdelni str",
      "zip_code": "1407",
      "city": "Rome",
      "country": "IT"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>ppro</transaction_type>
    <transaction_id>43671</transaction_id>
    <payment_type>mybank</payment_type>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <return_success_url>http://www.example.com/success</return_success_url>
    <return_failure_url>http://www.example.com/failure</return_failure_url>
    <amount>100</amount>
    <currency>EUR</currency>
    <billing_address>
      <first_name>Barney</first_name>
      <last_name>Rubble</last_name>
      <address1>14, Nerazdelni str</address1>
      <zip_code>1407</zip_code>
      <city>Rome</city>
      <country>IT</country>
    </billing_address>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | **ppro** or **my_bank**. Contact tech support at tech-support@e-comprocessing.com for more details. |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| payment_type | required[1] | ppro | MyBank. Contact tech support at tech-support@e-comprocessing.com for more details. |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |

| Parameter | Required | Format | Description |
|---|---|---|---|
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

`1 - payment_type must be submitted only when the transaction type is set to ppro`

**Supported currencies and countries**

| Currency code | Country code | Description |
|---|---|---|
| EUR | IT | for PPRO transaction type |
| EUR | IT, ES | for MyBank transaction type |

Successful Response

```
stdClass Object
(
    [transaction_type] => ppro
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:19.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => EUR
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[ppro]>
    <status content=[pending_async]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <redirect_url content=[https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61]>
    <timestamp content=[2020-07-15T12:08:19Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[EUR]>
    <sent_to_acquirer content=[true]>
]>
```

```
{
    transaction_type: "ppro",
    status: "pending_async",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    redirect_url: "https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61",
    timestamp: "2020-07-15T12:08:19Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "EUR",
    sent_to_acquirer: "true",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>ppro</transaction_type>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
  <timestamp>2020-07-15T12:08:19Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>EUR</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```


**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response


```
stdClass Object
(
    [transaction_type] => ppro
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 110
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:19.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => EUR
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[ppro]>
    <status content=[error]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <code content=[110]>
    <message content=[Something went wrong, please contact support!]>
    <timestamp content=[2020-07-15T12:08:19Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[EUR]>
    <sent_to_acquirer content=[true]>
]>
```

```
{
    transaction_type: "ppro",
    status: "error",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    code: "110",
    message: "Something went wrong, please contact support!",
    timestamp: "2020-07-15T12:08:19Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "EUR",
    sent_to_acquirer: "true",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>ppro</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>110</code>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:19Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>EUR</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

#### ONLINE BANKING

Online Banking is an oBeP-style alternative payment method that allows you to pay directly with your ebank account. After initiating a transaction, the online banking will redirect you to their page. There you will find a list with available banks to finish the payment.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\OnlineBankingPayments\OnlineBanking\Payin');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setNotificationUrl('https://www.example.com/notification')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setAmount('50000')
        ->setCurrency('CNY')
        ->setCustomerEmail('travis@example.com')
        ->setBankCode('CITIC')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```curl
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>online_banking</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <notification_url>https://www.example.com/notification</notification_url>
    <return_success_url>http://www.example.com/success</return_success_url>
    <return_failure_url>http://www.example.com/failure</return_failure_url>
    <amount>50000</amount>
    <currency>CNY</currency>
    <customer_email>travis@example.com</customer_email>
    <bank_code>CITIC</bank_code>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>10178</zip_code>
      <city>Los Angeles</city>
      <state>CA</state>
      <country>US</country>
    </billing_address>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **online_banking** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |

| Parameter | Required | Format | Description |
|---|---|---|---|
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| customer_email | required* | e-mail address | Must contain valid e-mail of customer |
| customer_phone | required* | string(32) | Must contain valid phone number of customer |
| bank_code | required | bank code | Must contain Bank code |
| document_id | required* | sring(20) | Document ID of the consumer. See Document ID Parameter for more details. |
| payment_type | required* | string | The payment type describes the type of online banking used to process the transaction. Must contain one of the allowedPayment types, but they may vary based on the specific setup. If omitted, transaction will be processed with**online_banking** payment_type. |
| virtual_payment_address | required* | string | Virtual Payment Address (VPA) of the customer, format: someone@bank |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

**Supported currencies**

| Currency name | Currency code |
|---|---|
| Argentine Peso | ARS |
| China yen | CNY |
| Chilean Peso | CLP |
| Colombian Peso | COP |
| Indian rupee | INR |
| Indonesian rupiah | IDR |
| Malaysian ringgit | MYR |
| Paraguayan Guarani | PYG |
| Philippine peso | PHP |
| Singapore dollar | SGD |
| Thai baht | THB |
| Uruguayan Peso | UYU |
| Vietnamese dong | VND |

**PAYMENT TYPES**

| Payment Type Name | Payment Type Code |
|---|---|
| Online banking | online_banking |
| Quick payment | quick_payment |
| Qr payment | qr_payment |
| Netbanking | netbanking |
| Alipay QR | alipay_qr |

Please, contact tech-support@e-comprocessing.com to find out more about which payment type to use with your setup.

**BANK CODES**

Bank codes may vary based on the specific setup.

**For ARS currency:**

| Bank Name | Bank Code |
|---|---|
| DirectDebit | DB |

**For BRL currency:**

| Bank Name | Bank Code |
|---|---|
| Caixa | CA |

**For CLP currency:**

| Bank Name | Bank Code |
|---|---|
| Servipag | SP |

**For CNY currency**

| Bank Name | Bank Code |
|---|---|
| Agricultural Bank of China | ABC |
| Bank of Beijing | BOB |
| Bank of China | BOC |
| Bank of Communications | BOCO |
| China Construction Bank | CCB |
| Bank for economic construction | CCD |
| China Everbright Bank | CEB |
| Industrial Bank | CIB |
| China Merchants Bank | CMB |
| China Minsheng Bank | CMBC |
| China Citic Bank | CITIC |
| Industrial and Commercial Bank of China | ICBC |
| China Guangfa Bank | GDB |
| Huaxia Bank | HXB |
| Pingan Bank | PINGANBANK |
| China Postal Savings Bank | PSBC |
| China Union Pay | QUICKPAY |
| Shanghai Bank | SHB |
| Shengzhen Ping An Bank | SPABANK |
| Shanghai Pudong Development Bank | SPDB |
| Yinlian Bank | YLB |

**For COP currency:**

| Bank Name | Bank Code |
|---|---|
| Bancolombia | PC |

**For IDR currency:**

| Bank Name | Bank Code |
|---|---|

| Bank Name | Bank Code |
|---|---|
| ALTO, Prima, ATM Bersama | ATMVA |
| Bank Central Asia | BCA_IDR |
| Bank Rakyat Indonesia | BRI_IDR |
| Bank Negara Indonesia | BNI_IDR |
| BTN Bank | BTN_IDR |
| CIMB Clicks Indonesia | CIMB_IDR |
| Danamon Bank | DMN_IDR |
| Mandiri Bank | MDR_IDR |
| Permata Bank | PMB_IDR |
| Virtual Account Bank | VA |

**For INR currency:**

The following bank codes can be combined with the**Online banking** payment type **only**.

| Bank Name | Bank Code |
|---|---|
| Netbanking | NB |

The following bank codes can be combined with the**Netbanking** payment type **only**.

| Bank Name | Bank Code |
|---|---|
| Aditya Birla Idea Payments Bank | ABPB |
| Airtel Payments Bank | AIRP |
| Allahabad Bank | ALLA |
| Andhra Bank | ANDB |
| Bank of Baroda - Retail Banking | BARB_R |
| Bank of Bahrein and Kuwait | BBKM |
| Dena Bank | BKDN |
| Bank of India | BKID |
| Central Bank of India | CBIN |
| City Union Bank | CIUB |
| Canara Bank | CNRB |
| Corporation Bank | CORP |
| Cosmos Co-operative Bank | COSB |
| Catholic Syrian Bank | CSBK |
| Development Bank of Singapore | DBSS |
| DCB Bank | DCBL |
| Deutsche Bank | DEUT |
| Dhanlaxmi Bank | DLXB |
| Equitas Small Finance Bank | ESFB |
| Federal Bank | FDRL |
| HDFC Bank | HDFC |
| IDBI | IBKL |
| ICICI Bank | ICIC |
| IDFC FIRST Bank | IDFB |

| Bank Name | Bank Code |
|---|---|
| Indian Bank | IDIB |
| Indusind Bank | INDB |
| Indian Overseas Bank | IOBA |
| Jammu and Kashmir Bank | JAKA |
| Janata Sahakari Bank (Pune) | JSBP |
| Karnataka Bank | KARB |
| Kotak Mahindra Bank | KKBK |
| Karur Vysya Bank | KVBL |
| Lakshmi Vilas Bank - Corporate Banking | LAVB_C |
| Lakshmi Vilas Bank - Retail Banking | LAVB_R |
| Bank of Maharashtra | MAHB |
| NKGSB Co-operative Bank | NKGS |
| Oriental Bank of Commerce | ORBC |
| Punjab & Maharashtra Co-operative Bank | PMCB |
| Punjab & Sind Bank | PSIB |
| Punjab National Bank - Retail Banking | PUNB_R |
| RBL Bank | RATN |
| State Bank of Bikaner and Jaipur | SBBJ |
| State Bank of Hyderabad | SBHY |
| State Bank of India | SBIN |
| State Bank of Mysore | SBMY |
| State Bank of Travancore | SBTR |
| Standard Chartered Bank | SCBL |
| South Indian Bank | SIBL |
| Saraswat Co-operative Bank | SRCB |
| State Bank of Patiala | STBP |
| Shamrao Vithal Co-operative Bank | SVCB |
| Syndicate Bank | SYNB |
| Tamilnadu Mercantile Bank | TMBL |
| Tamilnadu State Apex Co-operative Bank | TNSC |
| Union Bank of India | UBIN |
| UCO Bank | UCBA |
| United Bank of India | UTBI |
| Axis Bank | UTIB |
| Vijaya Bank | VIJB |
| Yes Bank | YESB |

**For MXN currency:**

| Bank Name | Bank Code |
|---|---|
| Spei | SE |
| Banorte | BQ |

**For MYR currency:**

| Bank Name | Bank Code |
|---|---|

| Bank Name | Bank Code |
|---|---|
| 7-eleven stores | CASH-711 |
| Affin Bank | FPX_ABB, AFFIN-EPG |
| Alliance Bank | FPX_ABMB |
| Am Online | FPX_AMB, AMB-W2W |
| Bank Islam | FPX_BIMB, BANKISLAM |
| Bank Muamalat | FPX_BMMB |
| Bank Rakyat | FPX_BKRM |
| Bank Simpanan Nasional | FPX_BSN |
| CIMB Clicks Bank | CIMB_MYR, FPX_CIMBCLICKS, CIMB-CLICKS |
| FPX | FPX |
| HLB Connect | FPX_HLB, HLB-ONL |
| Hong Leong Bank | HLE_MYR |
| jmpay | jompay.php |
| Kuwait Finance House | FPX_KFN |
| May Bank | MAY_MYR |
| Maybank2u | FPX_MB2U, MB2U |
| OCBC Bank | FPX_OCBC |
| PBeBank | FPX_PBB, PUBLICKBANK |
| Public Bank | PBE_MYR |
| RHB Bank | RHB_MYR |
| RHB Now | FPX_RHB, RHB-ONL |
| Stand Chart Bank | FPX_SCB |
| UOB Bank | FPX_UOB |

**For PEN currency:**

| Bank Name | Bank Code |
|---|---|
| BCP | BC |
| Interbank | IB |
| Pago Efectivo | EF |
| BBVA | BP |

**For PYG currency:**

| Bank Name | Bank Code |
|---|---|
| PagoExpress | PE |

**For THB currency:**

| Bank Name | Bank Code |
|---|---|
| Bangkok Bank | BBL_THB, TH_PB_BBLPN |
| Kasikorn Bank | KKB_THB |
| Krungsri (Bank of Ayudhya Public Company Limited) | BAY_THB, TH_PB_BAYPN |
| Krung Thai Bank | KTB_THB, TH_PB_KTBPN |
| OMISE_TL | OMISE_TL.php |
| Siam Commercial Bank | SCB_THB, TH_PB_SCBPN |
| UOBT | UOB_THB |

**For UYU currency:**

| Bank Name | Bank Code |
|---|---|
| Abitab | AI |

**For PHP currency:**

| Bank Name | Bank Code |
|---|---|
| Dragonpay | DRAGONPAY |

**For SGD currency:**

| Bank Name | Bank Code |
|---|---|
| SG eNETS | ENETS-D |
| singpost | singpost.php |

**For VND currency:**

| Bank Name | Bank Code |
|---|---|
| VTC-Pay | VTCP_VPBANK |
| VTC-Pay ABBANK | VTCP_ABBANK |
| VTC-Pay ACB | VTCP_ACB |
| VTC-Pay Agribank | VTCP_AGRIBANK |
| VTC-Pay BACABANK | VTCP_BACABANK |
| VTC-Pay BIDV | VTCP_BIDV |
| VTC-Pay BVB | VTCP_BVB |
| VTC-Pay DongABank | VTCP_DONGABANK |
| VTC-Pay Eximbank | VTCP_EXIMBANK |
| VTC-Pay GPBank | VTCP_GPBANK |
| VTC-Pay HDBank | VTCP_HDBANK |
| VTC-Pay LienVietPostBank | VTCP_LVPB |
| VTC-Pay MB | VTCP_MB |
| VTC-Pay MaritimeBank | VTCP_MARITIMEBANK |
| VTC-Pay NamABank | VTCP_NAMABANK |
| VTC-Pay Navibank | VTCP_NAVIBANK |
| VTC-Pay Oceanbank | VTCP_OCEANBANK |
| VTC-Pay PGBank | VTCP_PGBANK |
| VTC-Pay PHUONGDONG | VTCP_PHUONGDONG |
| VTC-Pay SHB | VTCP_SHB |
| VTC-Pay Sacombank | VTCP_SACOMBANK |
| VTC-Pay SaigonBank | VTCP-SAIGON_BANK |
| VTC-Pay SeaABank | VTCP_SEAABANK |
| VTC-Pay Techcombank | VTCP_TECHCOMBANK |
| VTC-Pay TienPhong Bank | VTCP_TIENPHONGBANK |
| VTC-Pay VIB | VTCP_VIB |
| VTC-Pay VietABank | VTCP_VIETABANK |
| VTC-Pay Vietcombank | VTCP_VIETCOMBANK |
| VTC-Pay Vietinbank | VTCP_VIETINBANK |

Successful Response

```
stdClass Object
(
    [transaction_type] => online_banking
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:19.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 50000
    [currency] => CNY
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>online_banking</transaction_type>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
  <timestamp>2020-07-15T12:08:19Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>50000</amount>
  <currency>CNY</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [transaction_type] => online_banking
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 110
    [technical_message] => amount is missing
    [message] => Please check input data for errors!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:19.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 50000
    [currency] => CNY
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>online_banking</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>110</code>
  <technical_message>amount is missing</technical_message>
  <message>Please check input data for errors!</message>
  <timestamp>2020-07-15T12:08:19Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>50000</amount>
  <currency>CNY</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

**P24**

P24 transactions are only asynchronous. After a successful validation of transaction parameters, transaction status is set to pending async and the user is redirected to the P24 payment page where he enters additional information to finish the payment. When the payment reaches a final state, Genesis gateway sends a notification to the merchant.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Alternatives\P24');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.apm.P24Request;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;
import java.net.MalformedURLException;
import java.net.URL;

public class GenesisExample {
  public static void main() throws MalformedURLException {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    P24Request request = new P24Request();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setReturnSuccessUrl(new URL("http://www.example.com/success"));
    request.setReturnFailureUrl(new URL("http://www.example.com/failure"));
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.p24(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "return_success_url": "http://www.example.com/success",
    "return_failure_url": "http://www.example.com/failure",
    "amount": "100",
    "currency": "USD",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>p24</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <return_success_url>http://www.example.com/success</return_success_url>
    <return_failure_url>http://www.example.com/failure</return_failure_url>
    <amount>100</amount>
    <currency>USD</currency>
    <customer_email>travis@example.com</customer_email>
    <customer_phone>+1987987987987</customer_phone>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>10178</zip_code>
      <city>Los Angeles</city>
      <state>CA</state>
      <country>US</country>
    </billing_address>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **p24** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | required* | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| customer_email | required | e-mail address | Must contain valid e-mail of customer |
| customer_phone | required* | string(32) | Must contain valid phone number of customer |
| **billing_address** | required | | See Required vs Optional API params for details |

| Parameter | Required | Format | Description |
|---|---|---|---|
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

Successful Response

```
stdClass Object
(
    [transaction_type] => p24
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:19.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[p24]>
    <status content=[pending_async]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <redirect_url content=[https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61]>
    <timestamp content=[2020-07-15T12:08:19Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[true]>
]>
```

```
{
    transaction_type: "p24",
    status: "pending_async",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    redirect_url: "https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61",
    timestamp: "2020-07-15T12:08:19Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "true",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>p24</transaction_type>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
  <timestamp>2020-07-15T12:08:19Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```


**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response


```
stdClass Object
(
    [transaction_type] => p24
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 110
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:19.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```xml
<payment_response content=[
    <transaction_type content=[p24]>
    <status content=[error]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <code content=[110]>
    <message content=[Something went wrong, please contact support!]>
    <timestamp content=[2020-07-15T12:08:19Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[true]>
]>
```

```json
{
    transaction_type: "p24",
    status: "error",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    code: "110",
    message: "Something went wrong, please contact support!",
    timestamp: "2020-07-15T12:08:19Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "true",
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>p24</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>110</code>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:19Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

#### PAYU

PayU is a payment method for Czech Republic and Poland

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\OnlineBankingPayments\PayU');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setAmount('100')
        ->setCurrency('PLN')
        ->setCustomerEmail('travis@example.com')


        // Billing Address
        ->setBillingFirstName('Barney')
        ->setBillingLastName('Rubble')
        ->setBillingAddress1('14, Nerazdelni str')
        ->setBillingZipCode('1407')
        ->setBillingCity('Krakov')
        ->setBillingCountry('PL')


        // Risk Params
        ->setRiskUserId('123456');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```curl
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>payu</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <return_success_url>http://www.example.com/success</return_success_url>
    <return_failure_url>http://www.example.com/failure</return_failure_url>
    <amount>100</amount>
    <currency>PLN</currency>
    <customer_email>travis@example.com</customer_email>
    <billing_address>
      <first_name>Barney</first_name>
      <last_name>Rubble</last_name>
      <address1>14, Nerazdelni str</address1>
      <zip_code>1407</zip_code>
      <city>Krakov</city>
      <country>PL</country>
    </billing_address>
    <risk_params>
      <user_id>123456</user_id>
    </risk_params>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **payu** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |

| Parameter | Required | Format | Description |
|---|---|---|---|
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| customer_email | optional | e-mail address | Must contain valid e-mail of customer |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required | string(255) | Customer first name |
| last_name | required | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

**Supported currencies and countries:**

| Currency code | Country code |
|---|---|
| CZK | CZ |
| PLN | PL |

Successful Response

```
stdClass Object
(
    [transaction_type] => payu
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:19.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => PLN
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>payu</transaction_type>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
  <timestamp>2020-07-15T12:08:19Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>PLN</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```


**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response


```
stdClass Object
(
    [transaction_type] => payu
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 110
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:19.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => PLN
    [sent_to_acquirer] => true
)
```


```
This request is not implemented yet
```

```
This request is not implemented yet
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>payu</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>110</code>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:19Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>PLN</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

**POLI**

POLi payment is Australia's most popular online banking. The payment method is available within Australia and New Zealand. POLi transactions are asynchronous. After successful validation of transaction parameters, transaction status is set to pending async, the user is redirected to POLi authentication page where he enters additional information to finish the payment. When the payment reaches a final state Genesis gateway sends a notification to the merchant on the URL sent in the request or the URL configured in their account.

Note, in some rare cases, the POLi system might not be able to confirm whether the user's payment is successful. In this case, the funds may have been transferred but the user will not be displayed a receipt from POLi or the merchant. This status can arise due to a bank issue or a local issue to the user.

**Therefore, POLi payments recommendation is for merchants to ensure that when end-user clicks on "Return to merchant's website", they land on a page that displays a clear message asking them to check their bank account before processing another transaction to make sure funds have not left the account.**

This will reduce the chances of duplicate transactions.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Alternatives\POLi');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setNotificationUrl('https://www.example.com/notification')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setAmount('100')
        ->setCurrency('AUD')


        // Billing Address
        ->setBillingFirstName('Barney')
        ->setBillingLastName('Rubble')
        ->setBillingAddress1('14, Nerazdelni str')
        ->setBillingZipCode('1407')
        ->setBillingCity('Sofia')
        ->setBillingCountry('AU')
        ->setBillingState('AC');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.apm.POLiRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;
import java.net.MalformedURLException;
import java.net.URL;

public class GenesisExample {
  public static void main() throws MalformedURLException {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    POLiRequest request = new POLiRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setNotificationUrl(new URL("https://www.example.com/notification"));
    request.setReturnSuccessUrl(new URL("http://www.example.com/success"));
    request.setReturnFailureUrl(new URL("http://www.example.com/failure"));
    request.setAmount(new BigDecimal(100));
    request.setCurrency("AUD");


    // Billing Address
    request.setBillingFirstname("Barney");
    request.setBillingLastname("Rubble");
    request.setBillingPrimaryAddress("14, Nerazdelni str");
    request.setBillingZipCode("1407");
    request.setBillingCity("Sofia");
    request.setBillingCountry("AU");
    request.setBillingState("AC");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.poli(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "notification_url": "https://www.example.com/notification",
    "return_success_url": "http://www.example.com/success",
    "return_failure_url": "http://www.example.com/failure",
    "amount": "100",
    "currency": "AUD",
    "billing_address": {
      "first_name": "Barney",
      "last_name": "Rubble",
      "address1": "14, Nerazdelni str",
      "zip_code": "1407",
      "city": "Sofia",
      "country": "AU",
      "state": "AC"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>poli</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <notification_url>https://www.example.com/notification</notification_url>
    <return_success_url>http://www.example.com/success</return_success_url>
    <return_failure_url>http://www.example.com/failure</return_failure_url>
    <amount>100</amount>
    <currency>AUD</currency>
    <billing_address>
      <first_name>Barney</first_name>
      <last_name>Rubble</last_name>
      <address1>14, Nerazdelni str</address1>
      <zip_code>1407</zip_code>
      <city>Sofia</city>
      <country>AU</country>
      <state>AC</state>
    </billing_address>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **poli** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| customer_email | required | e-mail address | Must contain valid e-mail of customer |
| customer_phone | required* | string(32) | Must contain valid phone number of customer |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |

| Parameter | Required | Format | Description |
|---|---|---|---|
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

**Supported bank countries:**

| Country name | Country code |
|---|---|
| Australia | AU |
| New Zealand | NZ |

**Supported currencies:**

| Currency name | Currency code |
|---|---|
| Australian dollar | AUD |
| New Zealand dollar | NZD |

Successful Response

```
stdClass Object
(
    [transaction_type] => poli
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:19.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => AUD
    [sent_to_acquirer] => true
)
```

```xml
<payment_response content=[
    <transaction_type content=[poli]>
    <status content=[pending_async]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <redirect_url content=[https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61]>
    <timestamp content=[2020-07-15T12:08:19Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[AUD]>
    <sent_to_acquirer content=[true]>
]>
```

```json
{
    transaction_type: "poli",
    status: "pending_async",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    redirect_url: "https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61",
    timestamp: "2020-07-15T12:08:19Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "AUD",
    sent_to_acquirer: "true",
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>poli</transaction_type>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
  <timestamp>2020-07-15T12:08:19Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>AUD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [transaction_type] => poli
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 110
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:19.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => AUD
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[poli]>
    <status content=[error]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <code content=[110]>
    <message content=[Something went wrong, please contact support!]>
    <timestamp content=[2020-07-15T12:08:19Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[AUD]>
    <sent_to_acquirer content=[true]>
]>
```

```
{
    transaction_type: "poli",
    status: "error",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    code: "110",
    message: "Something went wrong, please contact support!",
    timestamp: "2020-07-15T12:08:19Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "AUD",
    sent_to_acquirer: "true",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>poli</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>110</code>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:19Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>AUD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |

| Parameter | Type | Description |
|---|---|---|
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

**PSE (PAGOS SEGUROS EN LINEA)**

PSE (Pagos Seguros en Linea) is the preferred alternative payment solution in Colombia.The solution consists of an interface that offers the client the option to pay for their online purchases in cash, directing it to their online banking.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\OnlineBankingPayments\Pse');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setConsumerReference('barney_rumble')
        ->setNationalId('8812128812')
        ->setBirthDate('30-12-1992')


        // Billing Address
        ->setBillingFirstName('Barney')
        ->setBillingLastName('Rubble')
        ->setBillingAddress1('14, Nerazdelni str')
        ->setBillingZipCode('1407')
        ->setBillingCity('Bogota')
        ->setBillingCountry('CO')


        // Risk Params
        ->setRiskUserId('123456');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>pse</transaction_type>
   <transaction_id>43671</transaction_id>
   <usage>40208 concert tickets</usage>
   <remote_ip>245.253.2.12</remote_ip>
   <return_success_url>http://www.example.com/success</return_success_url>
   <return_failure_url>http://www.example.com/failure</return_failure_url>
   <amount>100</amount>
   <currency>USD</currency>
   <consumer_reference>barney_rumble</consumer_reference>
   <national_id>8812128812</national_id>
   <birth_date>30-12-1992</birth_date>
   <billing_address>
     <first_name>Barney</first_name>
     <last_name>Rubble</last_name>
     <address1>14, Nerazdelni str</address1>
     <zip_code>1407</zip_code>
     <city>Bogota</city>
     <country>CO</country>
   </billing_address>
   <risk_params>
     <user_id>123456</user_id>
   </risk_params>
 </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **pse** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| consumer_reference | required | string(20) | Consumer reference is a unique consumer identifier |
| national_id | required | string(20) | National ID of the consumer. See Document ID Parameter for more details. |
| birth_date | optional | dd-mm-yyyy | Required for Visa only when MCC is a Financial Services one (e.g. MCC 6012) |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |

| Parameter | Required | Format | Description |
|---|---|---|---|
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

**Supported countries:**

| Country |
|---|
| CO |

Successful Response

```
stdClass Object
(
    [transaction_type] => pse
    [status] => pending_async
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:19.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>pse</transaction_type>
  <status>pending_async</status>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <timestamp>2020-07-15T12:08:19Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |

| Parameter | Type | Description |
|---|---|---|
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [transaction_type] => pse
    [status] => error
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 110
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:19.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>pse</transaction_type>
  <status>error</status>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>110</code>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:19Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

**RAPIPAGO**

RapiPago from Argentina is an offline payment method used for online purchases. Shoppers buy their goods and services online and pay offline at one of the 6,000+ RapiPago payment locations.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\OnlineBankingPayments\Rapipago');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setConsumerReference('barney_rumble')
        ->setNationalId('8812128812')
        ->setBirthDate('30-12-1992')


        // Billing Address
        ->setBillingFirstName('Barney')
        ->setBillingLastName('Rubble')
        ->setBillingAddress1('14, Nerazdelni str')
        ->setBillingZipCode('1407')
        ->setBillingCity('Buenos Aires')
        ->setBillingCountry('AR')


        // Risk Params
        ->setRiskUserId('123456');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```curl
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>rapi_pago</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <return_success_url>http://www.example.com/success</return_success_url>
    <return_failure_url>http://www.example.com/failure</return_failure_url>
    <amount>100</amount>
    <currency>USD</currency>
    <consumer_reference>barney_rumble</consumer_reference>
    <national_id>8812128812</national_id>
    <birth_date>30-12-1992</birth_date>
    <billing_address>
      <first_name>Barney</first_name>
      <last_name>Rubble</last_name>
      <address1>14, Nerazdelni str</address1>
      <zip_code>1407</zip_code>
      <city>Buenos Aires</city>
      <country>AR</country>
    </billing_address>
    <risk_params>
      <user_id>123456</user_id>
    </risk_params>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **rapi_pago** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |

| Parameter | Required | Format | Description |
|---|---|---|---|
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| consumer_reference | required | string(20) | Consumer reference is a unique consumer identifier |
| national_id | required | string(20) | National ID of the consumer. See Document ID Parameter for more details. |
| birth_date | optional | dd-mm-yyyy | Required for Visa only when MCC is a Financial Services one (e.g. MCC 6012) |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

**Supported countries:**

| Country |
|---|
| AR |

Successful Response

```
stdClass Object
(
    [transaction_type] => rapi_pago
    [status] => pending_async
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:19.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```


```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>rapi_pago</transaction_type>
  <status>pending_async</status>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <timestamp>2020-07-15T12:08:19Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```


**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response


```
stdClass Object
(
    [transaction_type] => rapi_pago
    [status] => error
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 110
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:19.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```


```
This request is not implemented yet
```

```
This request is not implemented yet
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>rapi_pago</transaction_type>
  <status>error</status>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>110</code>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:19Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

**SAFETYPAY**

Safetypay is a real-time bank transfer system that operates in more than 10 different countries. Their main market is in Latin America.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\OnlineBankingPayments\SafetyPay');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setPaymentType('safetypay')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setConsumerReference('')
        ->setNationalId('')
        ->setBirthDate('')
        ->setCustomerEmail('travis@example.com')


        // Billing Address
        ->setBillingFirstName('Barney')
        ->setBillingLastName('Rubble')
        ->setBillingAddress1('14, Nerazdelni str')
        ->setBillingZipCode('1407')
        ->setBillingCity('Tampico')
        ->setBillingCountry('MX')


        // Risk Params
        ->setRiskUserId('123456');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```bash
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>safetypay</transaction_type>
    <transaction_id>43671</transaction_id>
    <payment_type>safetypay</payment_type>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <return_success_url>http://www.example.com/success</return_success_url>
    <return_failure_url>http://www.example.com/failure</return_failure_url>
    <amount>100</amount>
    <currency>USD</currency>
    <payment_transaction>consumer_reference</payment_transaction>
    <payment_transaction>national_id</payment_transaction>
    <payment_transaction>birth_date</payment_transaction>
    <customer_email>travis@example.com</customer_email>
    <billing_address>
      <first_name>Barney</first_name>
      <last_name>Rubble</last_name>
      <address1>14, Nerazdelni str</address1>
      <zip_code>1407</zip_code>
      <city>Tampico</city>
      <country>MX</country>
    </billing_address>
    <risk_params>
      <user_id>123456</user_id>
    </risk_params>
  </payment_transaction>'
```

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Alternatives\PPRO');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setPaymentType('safetypay')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setConsumerReference('')
        ->setNationalId('')
        ->setBirthDate('')
        ->setCustomerEmail('travis@example.com')


        // Billing Address
        ->setBillingFirstName('Barney')
        ->setBillingLastName('Rubble')
        ->setBillingAddress1('14, Nerazdelni str')
        ->setBillingZipCode('1407')
        ->setBillingCity('Tampico')
        ->setBillingCountry('MX')


        // Risk Params
        ->setRiskUserId('123456');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.apm.PProRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;
import java.net.MalformedURLException;
import java.net.URL;

public class GenesisExample {
  public static void main() throws MalformedURLException {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    PProRequest request = new PProRequest();

    request.setTransactionId("43671");
    request.setPaymentType("safetypay");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setReturnSuccessUrl(new URL("http://www.example.com/success"));
    request.setReturnFailureUrl(new URL("http://www.example.com/failure"));
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setConsumerReference("");
    request.setNationalId("");
    request.setBirthDate("");
    request.setCustomerEmail("travis@example.com");


    // Billing Address
    request.setBillingFirstname("Barney");
    request.setBillingLastname("Rubble");
    request.setBillingPrimaryAddress("14, Nerazdelni str");
    request.setBillingZipCode("1407");
    request.setBillingCity("Tampico");
    request.setBillingCountry("MX");


    // Risk Params
    request.setRiskUserId("123456");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.ppro(
  {
    "transaction_id": "43671",
    "payment_type": "safetypay",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "return_success_url": "http://www.example.com/success",
    "return_failure_url": "http://www.example.com/failure",
    "amount": "100",
    "currency": "USD",
    "consumer_reference": null,
    "national_id": null,
    "birth_date": null,
    "customer_email": "travis@example.com",
    "billing_address": {
      "first_name": "Barney",
      "last_name": "Rubble",
      "address1": "14, Nerazdelni str",
      "zip_code": "1407",
      "city": "Tampico",
      "country": "MX"
    },
    "risk_params": {
      "user_id": "123456"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>ppro</transaction_type>
    <transaction_id>43671</transaction_id>
    <payment_type>safetypay</payment_type>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <return_success_url>http://www.example.com/success</return_success_url>
    <return_failure_url>http://www.example.com/failure</return_failure_url>
    <amount>100</amount>
    <currency>USD</currency>
    <payment_transaction>consumer_reference</payment_transaction>
    <payment_transaction>national_id</payment_transaction>
    <payment_transaction>birth_date</payment_transaction>
    <customer_email>travis@example.com</customer_email>
    <billing_address>
      <first_name>Barney</first_name>
      <last_name>Rubble</last_name>
      <address1>14, Nerazdelni str</address1>
      <zip_code>1407</zip_code>
      <city>Tampico</city>
      <country>MX</country>
    </billing_address>
    <risk_params>
      <user_id>123456</user_id>
    </risk_params>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | **ppro** or **safetypay**. Contact tech support at tech-support@e-comprocessing.com for more details. |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| payment_type | required[1] | safetypay | SafetyPay. Contact tech support for more details |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |

| Parameter | Required | Format | Description |
|---|---|---|---|
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| customer_email | required | e-mail address | Must contain valid e-mail of customer |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

`1 - payment_type must be submitted only when the transaction type is set to ppro`

**Supported countries:**

| Country |
|---|
| AT |
| BE |
| BR |
| CL |
| CO |
| DE |
| EC |
| ES |
| MX |
| NL |
| PE |
| PR |

**Supported currencies:**

| Currency Code |
|---|
| EUR |
| USD |

Successful Response

```
stdClass Object
(
    [transaction_type] => safetypay
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:20.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>safetypay</transaction_type>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
  <timestamp>2020-07-15T12:08:20Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

Successful Response

```
stdClass Object
(
    [transaction_type] => ppro
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:20.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[ppro]>
    <status content=[pending_async]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <redirect_url content=[https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61]>
    <timestamp content=[2020-07-15T12:08:20Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[true]>
]>
```

```
{
    transaction_type: "ppro",
    status: "pending_async",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    redirect_url: "https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61",
    timestamp: "2020-07-15T12:08:20Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "true",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>ppro</transaction_type>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
  <timestamp>2020-07-15T12:08:20Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```


**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>ppro/safetypay</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>110</code>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:20Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

**SANTANDER**

Santander is an online bank transfer for ecommerce purchases. Consumers use their trusted home banking environment, merchants benefit from payment guarantee and swift settlement.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\OnlineBankingPayments\Santander');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setConsumerReference('barney_rubble')
        ->setNationalId('8812128812')
        ->setBirthDate('30-12-1992')
        ->setCustomerEmail('travis@example.com')


        // Billing Address
        ->setBillingFirstName('Barney')
        ->setBillingLastName('Rubble')
        ->setBillingAddress1('14, Nerazdelni str')
        ->setBillingZipCode('1407')
        ->setBillingCity('Rio de Janeiro')
        ->setBillingCountry('BR')


        // Risk Params
        ->setRiskUserId('123456');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```curl
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>santander</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <return_success_url>http://www.example.com/success</return_success_url>
    <return_failure_url>http://www.example.com/failure</return_failure_url>
    <amount>100</amount>
    <currency>USD</currency>
    <consumer_reference>barney_rubble</consumer_reference>
    <national_id>8812128812</national_id>
    <birth_date>30-12-1992</birth_date>
    <customer_email>travis@example.com</customer_email>
    <billing_address>
      <first_name>Barney</first_name>
      <last_name>Rubble</last_name>
      <address1>14, Nerazdelni str</address1>
      <zip_code>1407</zip_code>
      <city>Rio de Janeiro</city>
      <country>BR</country>
    </billing_address>
    <risk_params>
      <user_id>123456</user_id>
    </risk_params>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **santander** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |

| Parameter | Required | Format | Description |
|---|---|---|---|
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| consumer_reference | required | string(20) | Consumer reference is a unique consumer identifier |
| national_id | required | string(20) | National ID of the consumer. See Document ID Parameter for more details. |
| birth_date | optional | string(20) | Birth date of the customer |
| customer_email | required | e-mail address | Must contain valid e-mail of customer |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

**Supported countries:**

| Country |
|---|
| AR |
| BR |
| MX |
| CL |

Successful Response

```
stdClass Object
(
    [transaction_type] => santander
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:20.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>santander</transaction_type>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
  <timestamp>2020-07-15T12:08:20Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [transaction_type] => santander
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 110
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:20.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>santander</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>110</code>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:20Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

#### SDD INIT RECURRING SALE

An SddInitRecurringSale transaction initializes a recurring payment and is equal to a normalSddSaleTransaction except that it can be referenced as "initial" transaction in a SddRecurringSale transaction.

Note that if an SddInitRecurringSale is fully refunded, the recurring series is stopped and no more SddRecurringSales can be performed for that recurring series.

If an SddInitRecurringSale is partially refunded, the recurring series can continue with more SddRecurringSales.

> Authorize transactions are also available as 3dsecure transactions

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\SDD\Recurring\InitRecurringSale');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('EUR')
        ->setIban('DE09100100101234567891')
        ->setBic('PBNKDEFFXXX')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingCountry('DE');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.sdd.SDDInitRecurringSaleRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    SDDInitRecurringSaleRequest request = new SDDInitRecurringSaleRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("EUR");
    request.setIban("DE09100100101234567891");
    request.setBic("PBNKDEFFXXX");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingCountry("DE");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.sdd_init_recurring_sale(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "EUR",
    "iban": "DE09100100101234567891",
    "bic": "PBNKDEFFXXX",
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "country": "DE"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>sdd_init_recurring_sale</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>100</amount>
    <currency>EUR</currency>
    <iban>DE09100100101234567891</iban>
    <bic>PBNKDEFFXXX</bic>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <country>DE</country>
    </billing_address>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **sdd_init_recurring_sale** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | required | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| iban | required | string(34) | Customer's IBAN number |
| bic | optional | string(11) | SWIFT/BIC code of the customer's bank |
| customer_email | required* | e-mail address | Must contain valid e-mail of customer |
| customer_phone | required* | string(32) | Must contain valid phone number of customer |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required | string(255) | Customer first name |
| last_name | required | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |

| Parameter | Required | Format | Description |
|---|---|---|---|
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

**Supported countries:**

| Country Name | Country Code |
|---|---|
| Austria | AT |
| Belgium | BE |
| Cyprus | CY |
| Estonia | EE |
| Finland | FI |
| France | FR |
| Germany | DE |
| Greece | GR |
| Ireland | IE |
| Italy | IT |
| Latvia | LV |
| Lithuania | LT |
| Luxembourg | LU |
| Malta | MT |
| Monaco | MC |
| Netherlands | NL |
| Portugal | PT |
| Slovakia | SK |
| San Marino | SM |
| Slovenia | SI |
| Spain | ES |

Successful Response

```
stdClass Object
(
    [transaction_type] => sdd_init_recurring_sale
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:20.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => EUR
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[sdd_init_recurring_sale]>
    <status content=[pending_async]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <timestamp content=[2020-07-15T12:08:20Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[EUR]>
    <sent_to_acquirer content=[true]>
]>
```

```
{
    transaction_type: "sdd_init_recurring_sale",
    status: "pending_async",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    timestamp: "2020-07-15T12:08:20Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "EUR",
    sent_to_acquirer: "true",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>sdd_init_recurring_sale</transaction_type>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <timestamp>2020-07-15T12:08:20Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>EUR</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |

| Parameter | Type | Description |
|---|---|---|
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [transaction_type] => sdd_init_recurring_sale
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 110
    [message] => Please check input data for errors!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:20.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => EUR
    [sent_to_acquirer] => false
)
```

```
<payment_response content=[
    <transaction_type content=[sdd_init_recurring_sale]>
    <status content=[error]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <code content=[110]>
    <message content=[Please check input data for errors!]>
    <timestamp content=[2020-07-15T12:08:20Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[EUR]>
    <sent_to_acquirer content=[false]>
]>
```

```
{
    transaction_type: "sdd_init_recurring_sale",
    status: "error",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    code: "110",
    message: "Please check input data for errors!",
    timestamp: "2020-07-15T12:08:20Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "EUR",
    sent_to_acquirer: "false",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>sdd_init_recurring_sale</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>110</code>
  <message>Please check input data for errors!</message>
  <timestamp>2020-07-15T12:08:20Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>EUR</currency>
  <sent_to_acquirer>false</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |

| Parameter | Type | Description |
|---|---|---|
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

#### SDD RECURRING SALE

A SddRecurringSale transaction is a ”repeated” transaction which follows and references a SddInitRecurringSale transaction.

The bank account data is omitted.

Note that SddRecurringSales can be partially or fully refunded if configuration allows it, and this will not stop the sdd recurring series.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\SDD\Recurring\RecurringSale');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('EUR')
        ->setReferenceId('2ee4287e67971380ef7f97d5743bb523');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.sdd.SDDRecurringSaleRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    SDDRecurringSaleRequest request = new SDDRecurringSaleRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("EUR");
    request.setReferenceId("2ee4287e67971380ef7f97d5743bb523");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.sdd_recurring_sale(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "EUR",
    "reference_id": "2ee4287e67971380ef7f97d5743bb523"
  }
).send()
  .then(success)
  .catch(failure);
```

```bash
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>sdd_recurring_sale</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>100</amount>
    <currency>EUR</currency>
    <reference_id>2ee4287e67971380ef7f97d5743bb523</reference_id>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **sdd_recurring_sale** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | required | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |

| Parameter | Required | Format | Description |
|---|---|---|---|
| reference_id | required | string(32) | Unique id returned by corresponding transaction |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |

`required* = conditionally required`

Successful Response

```
stdClass Object
(
    [transaction_type] => sdd_recurring_sale
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:20.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => EUR
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[sdd_recurring_sale]>
    <status content=[pending_async]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <timestamp content=[2020-07-15T12:08:20Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[EUR]>
    <sent_to_acquirer content=[true]>
]>
```

```
{
    transaction_type: "sdd_recurring_sale",
    status: "pending_async",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    timestamp: "2020-07-15T12:08:20Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "EUR",
    sent_to_acquirer: "true",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>sdd_recurring_sale</transaction_type>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <timestamp>2020-07-15T12:08:20Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>EUR</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |

| Parameter | Type | Description |
|---|---|---|
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [transaction_type] => sdd_recurring_sale
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 320
    [technical_message] => amount is missing!
    [message] => Please check input data for errors!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:20.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => EUR
)
```

```
<payment_response content=[
    <transaction_type content=[sdd_recurring_sale]>
    <status content=[error]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <code content=[320]>
    <technical_message content=[amount is missing!]>
    <message content=[Please check input data for errors!]>
    <timestamp content=[2020-07-15T12:08:20Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[EUR]>
]>
```

```
{
    transaction_type: "sdd_recurring_sale",
    status: "error",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    code: "320",
    technical_message: "amount is missing!",
    message: "Please check input data for errors!",
    timestamp: "2020-07-15T12:08:20Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "EUR",
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>sdd_recurring_sale</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>320</code>
  <technical_message>amount is missing!</technical_message>
  <message>Please check input data for errors!</message>
  <timestamp>2020-07-15T12:08:20Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>EUR</currency>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

**SDD REFUND**

**SddRefunds allow to return already billed amounts to customers.**

The amount can be fully refunded only, no partial refunds are allowed. Note that SDD refunds can only be done on former approved SDD transactions

Therefore, the reference_id for the corresponding transaction is mandatory.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\SDD\Refund');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setAmount('100')
        ->setCurrency('EUR')
        ->setReferenceId('2ee4287e67971380ef7f97d5743bb523');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.sdd.SDDRefundRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    SDDRefundRequest request = new SDDRefundRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("EUR");
    request.setReferenceId("2ee4287e67971380ef7f97d5743bb523");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```node
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.sdd_refund(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "amount": "100",
    "currency": "EUR",
    "reference_id": "2ee4287e67971380ef7f97d5743bb523"
  }
).send()
  .then(success)
  .catch(failure);
```

```curl
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>sdd_refund</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <amount>100</amount>
    <currency>EUR</currency>
    <reference_id>2ee4287e67971380ef7f97d5743bb523</reference_id>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **sdd_refund** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | required | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| reference_id | required | string(32) | Unique id returned by corresponding transaction |

`required* = conditionally required`

Successful Response

```
stdClass Object
(
    [transaction_type] => sdd_refund
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:20.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => EUR
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[sdd_refund]>
    <status content=[pending_async]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <timestamp content=[2020-07-15T12:08:20Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[EUR]>
    <sent_to_acquirer content=[true]>
]>
```

```
{
    transaction_type: "sdd_refund",
    status: "pending_async",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    timestamp: "2020-07-15T12:08:20Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "EUR",
    sent_to_acquirer: "true",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>sdd_refund</transaction_type>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <timestamp>2020-07-15T12:08:20Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>EUR</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |

| Parameter | Type | Description |
|---|---|---|
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [transaction_type] => sdd_refund
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 320
    [technical_message] => amount is missing!
    [message] => Please check input data for errors!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:20.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => EUR
    [sent_to_acquirer] => false
)
```

```
<payment_response content=[
    <transaction_type content=[sdd_refund]>
    <status content=[error]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <code content=[320]>
    <technical_message content=[amount is missing!]>
    <message content=[Please check input data for errors!]>
    <timestamp content=[2020-07-15T12:08:20Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[EUR]>
    <sent_to_acquirer content=[false]>
]>
```

```
{
    transaction_type: "sdd_refund",
    status: "error",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    code: "320",
    technical_message: "amount is missing!",
    message: "Please check input data for errors!",
    timestamp: "2020-07-15T12:08:20Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "EUR",
    sent_to_acquirer: "false",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>sdd_refund</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>320</code>
  <technical_message>amount is missing!</technical_message>
  <message>Please check input data for errors!</message>
  <timestamp>2020-07-15T12:08:20Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>EUR</currency>
  <sent_to_acquirer>false</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

**SDD SALE**

The status of Sepa Direct Debit transactions is not available right after a transaction is made. Merchants receive the status of SDD transaction at 8:30 am (CET), 10:30 am (CET), 3:30 pm (CET) and 7:30 pm (CET). The merchant should have enabled notifications

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\SDD\Sale');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('EUR')
        ->setIban('DE09100100101234567891')
        ->setBic('PBNKDEFFXXX')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingCountry('DE');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.sdd.SDDSaleRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    SDDSaleRequest request = new SDDSaleRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("EUR");
    request.setIban("DE09100100101234567891");
    request.setBic("PBNKDEFFXXX");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingCountry("DE");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.sdd_sale(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "EUR",
    "iban": "DE09100100101234567891",
    "bic": "PBNKDEFFXXX",
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "country": "DE"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```xml
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>sdd_sale</transaction_type>
   <transaction_id>43671</transaction_id>
   <usage>40208 concert tickets</usage>
   <remote_ip>245.253.2.12</remote_ip>
   <amount>100</amount>
   <currency>EUR</currency>
   <iban>DE09100100101234567891</iban>
   <bic>PBNKDEFFXXX</bic>
   <billing_address>
     <first_name>Travis</first_name>
     <last_name>Pastrana</last_name>
     <country>DE</country>
   </billing_address>
 </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **sdd_sale** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | required | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| iban | required | string(34) | Customer's IBAN number |
| bic | optional | string(11) | SWIFT/BIC code of the customer's bank |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required | string(255) | Customer first name |
| last_name | required | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

**Supported countries: **

The supported countries are the same as SDD Init Recurring Sale.

Successful Response

```
stdClass Object
(
    [transaction_type] => sdd_sale
    [status] => approved
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:20.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => EUR
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[sdd_sale]>
    <status content=[approved]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <timestamp content=[2020-07-15T12:08:20Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[EUR]>
    <sent_to_acquirer content=[true]>
]>
```

```
{
    transaction_type: "sdd_sale",
    status: "approved",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    timestamp: "2020-07-15T12:08:20Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "EUR",
    sent_to_acquirer: "true",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>sdd_sale</transaction_type>
  <status>approved</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <timestamp>2020-07-15T12:08:20Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>EUR</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |

| Parameter | Type | Description |
|---|---|---|
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [transaction_type] => sdd_sale
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 340
    [message] => expiration_year is invalid
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:20.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => EUR
    [sent_to_acquirer] => false
)
```

```
<payment_response content=[
    <transaction_type content=[sdd_sale]>
    <status content=[error]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <code content=[340]>
    <message content=[expiration_year is invalid]>
    <timestamp content=[2020-07-15T12:08:20Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[EUR]>
    <sent_to_acquirer content=[false]>
]>
```

```
{
    transaction_type: "sdd_sale",
    status: "error",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    code: "340",
    message: "expiration_year is invalid",
    timestamp: "2020-07-15T12:08:20Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "EUR",
    sent_to_acquirer: "false",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>sdd_sale</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>340</code>
  <message>expiration_year is invalid</message>
  <timestamp>2020-07-15T12:08:20Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>EUR</currency>
  <sent_to_acquirer>false</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |

| Parameter | Type | Description |
|---|---|---|
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08- 30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |
| dynamic_descriptor_params | section | Optional, returned only if dynamic desc params are submitted on the API. Note here that the formatted dyn desc params are returned - as they would be submitted to the schemes for settlement. |

**SOFORT**

Sofort transactions are only asynchronous. After a successful validation of transaction parameters, transaction status is set to pending async, the user is redirected to Sofort authentication page where he enters additional information to finish the payment. When the payment reaches a final state Genesis gateway sends notification to merchant on the configured url into its account.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Alternatives\Sofort');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setAmount('100')
        ->setCurrency('EUR')
        ->setCustomerEmail('travis@example.com')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Berlin')
        ->setBillingCountry('DE');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.apm.SofortRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;
import java.net.MalformedURLException;
import java.net.URL;

public class GenesisExample {
  public static void main() throws MalformedURLException {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    SofortRequest request = new SofortRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setReturnSuccessUrl(new URL("http://www.example.com/success"));
    request.setReturnFailureUrl(new URL("http://www.example.com/failure"));
    request.setAmount(new BigDecimal(100));
    request.setCurrency("EUR");
    request.setCustomerEmail("travis@example.com");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Berlin");
    request.setBillingCountry("DE");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.sofort(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "return_success_url": "http://www.example.com/success",
    "return_failure_url": "http://www.example.com/failure",
    "amount": "100",
    "currency": "EUR",
    "customer_email": "travis@example.com",
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Berlin",
      "country": "DE"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>sofort</transaction_type>
   <transaction_id>43671</transaction_id>
   <usage>40208 concert tickets</usage>
   <remote_ip>245.253.2.12</remote_ip>
   <return_success_url>http://www.example.com/success</return_success_url>
   <return_failure_url>http://www.example.com/failure</return_failure_url>
   <amount>100</amount>
   <currency>EUR</currency>
   <customer_email>travis@example.com</customer_email>
   <billing_address>
     <first_name>Travis</first_name>
     <last_name>Pastrana</last_name>
     <address1>Muster Str. 12</address1>
     <zip_code>10178</zip_code>
     <city>Berlin</city>
     <country>DE</country>
   </billing_address>
 </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **sofort** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| customer_email | required | e-mail address | Must contain valid e-mail of customer |
| customer_phone | required* | string(32) | Must contain valid phone number of customer |
| bank_account_number | optional | string(24) | Bank identification number of the customer |
| bic | optional | string(12) | Bank Identifier Code |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |

| Parameter | Required | Format | Description |
|---|---|---|---|
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

**Supported countries:**

| Country name | Country code |
|---|---|
| Austria | AT |
| Belgium | BE |
| Germany | DE |
| Italy | IT |
| Netherlands | NL |
| Poland | PL |
| Spain | ES |
| Switzerland | CH |

Successful Response

```
stdClass Object
(
    [transaction_type] => sofort
    [status] => approved
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:20.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => EUR
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[sofort]>
    <status content=[approved]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <redirect_url content=[https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61]>
    <timestamp content=[2020-07-15T12:08:20Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[EUR]>
    <sent_to_acquirer content=[true]>
]>
```

```
{
    transaction_type: "sofort",
    status: "approved",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    redirect_url: "https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61",
    timestamp: "2020-07-15T12:08:20Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "EUR",
    sent_to_acquirer: "true",
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>sofort</transaction_type>
  <status>approved</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
  <timestamp>2020-07-15T12:08:20Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>EUR</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```


**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response


```
stdClass Object
(
    [transaction_type] => sofort
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 110
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:20.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => EUR
    [sent_to_acquirer] => true
)
```



```
<payment_response content=[
    <transaction_type content=[sofort]>
    <status content=[error]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <code content=[110]>
    <message content=[Something went wrong, please contact support!]>
    <timestamp content=[2020-07-15T12:08:20Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[EUR]>
    <sent_to_acquirer content=[true]>
]>
```

```
{
    transaction_type: "sofort",
    status: "error",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    code: "110",
    message: "Something went wrong, please contact support!",
    timestamp: "2020-07-15T12:08:20Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "EUR",
    sent_to_acquirer: "true",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>sofort</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>110</code>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:20Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>EUR</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```


**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

**TRUSTLY SALE**

Trustly is an oBeP-style alternative payment method that allows you to pay directly with your ebank account.

It has two transaction types sale and withdrawal. After initiating a transaction Trustly will redirect you to their page.

There you have to select your bank and log in with your regular access codes. Choose the account from which you wish to pay (for example, your savings account or current account).

When using your own hosted payment form, please follow Trustly requirements on services presentation and branding. Please contact your AM or our Technical team.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Alternatives\Trustly\Sale');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setPaymentType('')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setNotificationUrl('https://www.example.com/notification')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setAmount('100')
        ->setCurrency('EUR')
        ->setCustomerEmail('travis@example.com')


        // Billing Address
        ->setBillingFirstName('Max')
        ->setBillingLastName('Mustermann')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Berlin')
        ->setBillingCountry('DE');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.apm.TrustlySaleRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;
import java.net.MalformedURLException;
import java.net.URL;

public class GenesisExample {
  public static void main() throws MalformedURLException {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    TrustlySaleRequest request = new TrustlySaleRequest();

    request.setTransactionId("43671");
    request.setPaymentType("");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setNotificationUrl(new URL("https://www.example.com/notification"));
    request.setReturnSuccessUrl(new URL("http://www.example.com/success"));
    request.setReturnFailureUrl(new URL("http://www.example.com/failure"));
    request.setAmount(new BigDecimal(100));
    request.setCurrency("EUR");
    request.setCustomerEmail("travis@example.com");


    // Billing Address
    request.setBillingFirstname("Max");
    request.setBillingLastname("Mustermann");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Berlin");
    request.setBillingCountry("DE");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.trustly_sale(
  {
    "transaction_id": "43671",
    "payment_type": null,
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "notification_url": "https://www.example.com/notification",
    "return_success_url": "http://www.example.com/success",
    "return_failure_url": "http://www.example.com/failure",
    "amount": "100",
    "currency": "EUR",
    "customer_email": "travis@example.com",
    "billing_address": {
      "first_name": "Max",
      "last_name": "Mustermann",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Berlin",
      "country": "DE"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>trustly_sale</transaction_type>
    <transaction_id>43671</transaction_id>
    <payment_transaction>payment_type</payment_transaction>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <notification_url>https://www.example.com/notification</notification_url>
    <return_success_url>http://www.example.com/success</return_success_url>
    <return_failure_url>http://www.example.com/failure</return_failure_url>
    <amount>100</amount>
    <currency>EUR</currency>
    <customer_email>travis@example.com</customer_email>
    <billing_address>
      <first_name>Max</first_name>
      <last_name>Mustermann</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>10178</zip_code>
      <city>Berlin</city>
      <country>DE</country>
    </billing_address>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **trustly_sale** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| customer_email | required | e-mail address | Must contain valid e-mail of customer |
| customer_phone | required* | string(32) | Must contain valid phone number of customer |
| user_id | required* | string(255) | Unique user identifier defined by merchant |

| Parameter | Required | Format | Description |
|---|---|---|---|
| birth_date | optional | dd-mm-yyyy | Date of birth of the beneficiary, or organisational number for the organisation. |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

**Supported countries:**

| Country name | Country code |
|---|---|
| Austria | AT |
| Belgium | BE |
| Czech Republic | CZ |
| Denmark | DK |
| Estonia | EE |
| Finland | FI |
| Germany | DE |
| Latvia | LV |
| Lithuania | LT |
| Netherlands | NL |
| Norway | NO |
| Poland | PL |
| Slovakia | SK |
| Spain | ES |
| Sweden | SE |
| United Kingdom | GB |

Successful Response

```
stdClass Object
(
    [transaction_type] => trustly_sale
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:20.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => EUR
    [sent_to_acquirer] => true
)
```

```xml
<payment_response content=[
    <transaction_type content=[trustly_sale]>
    <status content=[pending_async]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <redirect_url content=[https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61]>
    <timestamp content=[2020-07-15T12:08:20Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[EUR]>
    <sent_to_acquirer content=[true]>
]>
```

```json
{
    transaction_type: "trustly_sale",
    status: "pending_async",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    redirect_url: "https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61",
    timestamp: "2020-07-15T12:08:20Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "EUR",
    sent_to_acquirer: "true",
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>trustly_sale</transaction_type>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
  <timestamp>2020-07-15T12:08:20Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>EUR</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [transaction_type] => trustly_sale
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 110
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:20.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => EUR
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[trustly_sale]>
    <status content=[error]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <code content=[110]>
    <message content=[Something went wrong, please contact support!]>
    <timestamp content=[2020-07-15T12:08:20Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[EUR]>
    <sent_to_acquirer content=[true]>
]>
```

```
{
    transaction_type: "trustly_sale",
    status: "error",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    code: "110",
    message: "Something went wrong, please contact support!",
    timestamp: "2020-07-15T12:08:20Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "EUR",
    sent_to_acquirer: "true",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>trustly_sale</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>110</code>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:20Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>EUR</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |

| Parameter | Type | Description |
|---|---|---|
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

**TRUSTPAY**

TrustPay is a real-time bank transfer payment service, which is widely used in the Czech Republic and Slovakia.

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>trustpay</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <return_success_url>http://www.example.com/success</return_success_url>
    <return_failure_url>http://www.example.com/failure</return_failure_url>
    <amount>100</amount>
    <currency>EUR</currency>
    <customer_phone>+1987987987987</customer_phone>
    <billing_address>
      <first_name>Barney</first_name>
      <last_name>Rubble</last_name>
      <address1>14, Nerazdelni str</address1>
      <zip_code>1407</zip_code>
      <city>Prague</city>
      <country>CZ</country>
    </billing_address>
  </payment_transaction>'
```

Request

```
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Alternatives\PPRO');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setPaymentType('trustpay')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setAmount('100')
        ->setCurrency('EUR')


        // Billing Address
        ->setBillingFirstName('Barney')
        ->setBillingLastName('Rubble')
        ->setBillingAddress1('14, Nerazdelni str')
        ->setBillingZipCode('1407')
        ->setBillingCity('Prague')
        ->setBillingCountry('CZ');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.apm.PProRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;
import java.net.MalformedURLException;
import java.net.URL;

public class GenesisExample {
  public static void main() throws MalformedURLException {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    PProRequest request = new PProRequest();

    request.setTransactionId("43671");
    request.setPaymentType("trustpay");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setReturnSuccessUrl(new URL("http://www.example.com/success"));
    request.setReturnFailureUrl(new URL("http://www.example.com/failure"));
    request.setAmount(new BigDecimal(100));
    request.setCurrency("EUR");


    // Billing Address
    request.setBillingFirstname("Barney");
    request.setBillingLastname("Rubble");
    request.setBillingPrimaryAddress("14, Nerazdelni str");
    request.setBillingZipCode("1407");
    request.setBillingCity("Prague");
    request.setBillingCountry("CZ");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.ppro(
  {
    "transaction_id": "43671",
    "payment_type": "trustpay",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "return_success_url": "http://www.example.com/success",
    "return_failure_url": "http://www.example.com/failure",
    "amount": "100",
    "currency": "EUR",
    "billing_address": {
      "first_name": "Barney",
      "last_name": "Rubble",
      "address1": "14, Nerazdelni str",
      "zip_code": "1407",
      "city": "Prague",
      "country": "CZ"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>ppro</transaction_type>
   <transaction_id>43671</transaction_id>
   <payment_type>trustpay</payment_type>
   <usage>40208 concert tickets</usage>
   <remote_ip>245.253.2.12</remote_ip>
   <return_success_url>http://www.example.com/success</return_success_url>
   <return_failure_url>http://www.example.com/failure</return_failure_url>
   <amount>100</amount>
   <currency>EUR</currency>
   <billing_address>
     <first_name>Barney</first_name>
     <last_name>Rubble</last_name>
     <address1>14, Nerazdelni str</address1>
     <zip_code>1407</zip_code>
     <city>Prague</city>
     <country>CZ</country>
   </billing_address>
 </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | **ppro** or **trustpay**. Contact tech support at tech-support@e-comprocessing.com for more details. |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| payment_type | required[1] | ppro | TrustPay. Contact tech support at tech-support@e-comprocessing.com for more details. |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

`1 - payment_type must be submitted only when the transaction type is set to ppro`

**Supported countries**

| Country code |
| --- |
| CZ |
| SK |

**Supported currencies**

| Currency code |
| --- |
| CZK |
| EUR |

Successful Response

```
stdClass Object
(
    [transaction_type] => ppro
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:20.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => EUR
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[ppro]>
    <status content=[pending_async]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <redirect_url content=[https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61]>
    <timestamp content=[2020-07-15T12:08:20Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[EUR]>
    <sent_to_acquirer content=[true]>
]>
```

```
{
    transaction_type: "ppro",
    status: "pending_async",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    redirect_url: "https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61",
    timestamp: "2020-07-15T12:08:20Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "EUR",
    sent_to_acquirer: "true",
}
```

```
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>ppro</transaction_type>
    <status>pending_async</status>
    <mode>live</mode>
    <transaction_id>43671</transaction_id>
    <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
    <technical_message>Transaction successful!</technical_message>
    <message>Transaction successful!</message>
    <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
    <timestamp>2020-07-15T12:08:20Z</timestamp>
    <descriptor>Descriptor one</descriptor>
    <amount>100</amount>
    <currency>EUR</currency>
    <sent_to_acquirer>true</sent_to_acquirer>
  </payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [transaction_type] => ppro
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 110
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:20.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => EUR
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[ppro]>
    <status content=[error]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <code content=[110]>
    <message content=[Something went wrong, please contact support!]>
    <timestamp content=[2020-07-15T12:08:20Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[EUR]>
    <sent_to_acquirer content=[true]>
]>
```

```
{
    transaction_type: "ppro",
    status: "error",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    code: "110",
    message: "Something went wrong, please contact support!",
    timestamp: "2020-07-15T12:08:20Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "EUR",
    sent_to_acquirer: "true",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>ppro</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>110</code>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:20Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>EUR</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

**UPI**

UPI (Unified Payment Interface) transaction is an alternative payment method which allows users to transfer money between bank accounts.

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>upi</transaction_type>
   <transaction_id>43671</transaction_id>
   <usage>40208 concert tickets</usage>
   <remote_ip>245.253.2.12</remote_ip>
   <notification_url>https://www.example.com/notification</notification_url>
   <return_success_url>http://www.example.com/success</return_success_url>
   <return_failure_url>http://www.example.com/failure</return_failure_url>
   <amount>50000</amount>
   <currency>INR</currency>
   <customer_email>travis@example.com</customer_email>
   <virtual_payment_address>someone@bank</virtual_payment_address>
   <billing_address>
     <first_name>Travis</first_name>
     <last_name>Pastrana</last_name>
     <address1>Muster Str. 12</address1>
     <zip_code>10178</zip_code>
     <city>Los Angeles</city>
     <state>CA</state>
     <country>US</country>
   </billing_address>
 </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **upi** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | required* | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| customer_email | required* | e-mail address | Must contain valid e-mail of customer |
| customer_phone | required* | string(32) | Must contain valid phone number of customer |
| document_id | required* | sring(20) | Document ID of the consumer. See Document ID Parameter for more details. |
| virtual_payment_address | required | string(255) | Virtual Payment Address (VPA) of the customer, format: someone@bank |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

Virtual payment address is used and required for Unified Payment Interface (UPI) transactions.

**Supported currencies**

| Currency name | Currency code |
|---|---|
| Indian rupee | INR |

Successful Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>upi</transaction_type>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
  <timestamp>2020-07-15T12:08:20Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>50000</amount>
  <currency>INR</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```


**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```


```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>upi</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>110</code>
  <technical_message>amount is missing</technical_message>
  <message>Please check input data for errors!</message>
  <timestamp>2020-07-15T12:08:20Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>50000</amount>
  <currency>INR</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```


**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |

| Parameter | Type | Description |
|---|---|---|
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

**WEBPAY**

Webpay is a Chilean real-time bank transfer method.

This transaction type is refundable via Refund transaction.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\OnlineBankingPayments\Webpay');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setConsumerReference('john_doe')
        ->setNationalId('8812128812')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Barney')
        ->setBillingLastName('Rubble')
        ->setBillingAddress1('14, Nerazdelni str')
        ->setBillingZipCode('1407')
        ->setBillingCity('Santiago')
        ->setBillingCountry('CL');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>webpay</transaction_type>
   <transaction_id>43671</transaction_id>
   <usage>40208 concert tickets</usage>
   <remote_ip>245.253.2.12</remote_ip>
   <return_success_url>http://www.example.com/success</return_success_url>
   <return_failure_url>http://www.example.com/failure</return_failure_url>
   <amount>100</amount>
   <currency>USD</currency>
   <consumer_reference>john_doe</consumer_reference>
   <national_id>8812128812</national_id>
   <customer_email>travis@example.com</customer_email>
   <customer_phone>+1987987987987</customer_phone>
   <billing_address>
     <first_name>Barney</first_name>
     <last_name>Rubble</last_name>
     <address1>14, Nerazdelni str</address1>
     <zip_code>1407</zip_code>
     <city>Santiago</city>
     <country>CL</country>
   </billing_address>
 </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **webpay** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| consumer_reference | required | string(20) | Consumer reference is a unique consumer identifier |
| national_id | required | string(20) | National ID of the consumer. See Document ID Parameter for more details. |
| birth_date | optional | dd-mm-yyyy | Required for Visa only when MCC is a Financial Services one (e.g. MCC 6012) |
| customer_email | required | e-mail address | Must contain valid e-mail of customer |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |

| Parameter | Required | Format | Description |
|---|---|---|---|
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

**Supported countries**

| Country Name | Country code |
|---|---|
| Chile | CL |

Successful Response

```
stdClass Object
(
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:20.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <timestamp>2020-07-15T12:08:20Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [status] => error
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 110
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:20.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => false
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <status>error</status>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>110</code>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:20Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>false</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

**DAVIVIENDA**

Davivienda is offering the Bill pay service which is a fast, easy and secure way to pay and manage your bills online to anyone, anytime in Colombia.

This transaction type is refundable via Refund transaction.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\OnlineBankingPayments\Davivienda');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setConsumerReference('barney_rumble')
        ->setNationalId('8812128812')
        ->setBirthDate('30-12-1992')
        ->setCustomerEmail('barney@example.com')


        // Billing Address
        ->setBillingFirstName('Barney')
        ->setBillingLastName('Rubble')
        ->setBillingAddress1('14, Nerazdelni str')
        ->setBillingZipCode('1407')
        ->setBillingCity('Bogota')
        ->setBillingCountry('CO')


        // Risk Params
        ->setRiskUserId('123456');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```curl
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>davivienda</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <return_success_url>http://www.example.com/success</return_success_url>
    <return_failure_url>http://www.example.com/failure</return_failure_url>
    <amount>100</amount>
    <currency>USD</currency>
    <consumer_reference>barney_rumble</consumer_reference>
    <national_id>8812128812</national_id>
    <birth_date>30-12-1992</birth_date>
    <customer_email>barney@example.com</customer_email>
    <billing_address>
      <first_name>Barney</first_name>
      <last_name>Rubble</last_name>
      <address1>14, Nerazdelni str</address1>
      <zip_code>1407</zip_code>
      <city>Bogota</city>
      <country>CO</country>
    </billing_address>
    <risk_params>
      <user_id>123456</user_id>
    </risk_params>
  </payment_transaction>'
```

### Request Parameters

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **davivienda** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |

| Parameter | Required | Format | Description |
|---|---|---|---|
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| consumer_reference | required | string(20) | Consumer reference is a unique consumer identifier |
| national_id | required | string(20) | National ID of the consumer. See Document ID Parameter for more details. |
| birth_date | optional | dd-mm-yyyy | Required for Visa only when MCC is a Financial Services one (e.g. MCC 6012) |
| customer_email | required | e-mail address | Must contain valid e-mail of customer |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

**Supported countries:**

| Country |
|---|
| CO |

Successful Response

```
stdClass Object
(
    [transaction_type] => davivienda
    [status] => pending_async
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:20.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```


```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>davivienda</transaction_type>
  <status>pending_async</status>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <timestamp>2020-07-15T12:08:20Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```


**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [transaction_type] => davivienda
    [status] => error
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 110
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:20.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>davivienda</transaction_type>
  <status>error</status>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>110</code>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:20Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

# Cash Payments

**BALOTO**

Baloto is a cash payment option in Colombia. It allows the customers to receive a voucher at check-out. The voucher can then be paid in any of the Via Boleto offices in cash.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\CashPayments\Baloto');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setConsumerReference('barney_rubble')
        ->setNationalId('8812128812')
        ->setBirthDate('30-12-1992')
        ->setCustomerEmail('travis@example.com')


        // Billing Address
        ->setBillingFirstName('Barney')
        ->setBillingLastName('Rubble')
        ->setBillingAddress1('14, Nerazdelni str')
        ->setBillingZipCode('1407')
        ->setBillingCity('Bogota')
        ->setBillingCountry('CO')


        // Risk Params
        ->setRiskUserId('123456');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```curl
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>baloto</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <return_success_url>http://www.example.com/success</return_success_url>
    <return_failure_url>http://www.example.com/failure</return_failure_url>
    <amount>100</amount>
    <currency>USD</currency>
    <consumer_reference>barney_rubble</consumer_reference>
    <national_id>8812128812</national_id>
    <birth_date>30-12-1992</birth_date>
    <customer_email>travis@example.com</customer_email>
    <billing_address>
      <first_name>Barney</first_name>
      <last_name>Rubble</last_name>
      <address1>14, Nerazdelni str</address1>
      <zip_code>1407</zip_code>
      <city>Bogota</city>
      <country>CO</country>
    </billing_address>
    <risk_params>
      <user_id>123456</user_id>
    </risk_params>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **baloto** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |

| Parameter | Required | Format | Description |
|---|---|---|---|
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| consumer_reference | required | string(20) | Consumer reference is a unique consumer identifier |
| national_id | required | string(20) | National ID of the consumer. See Document ID Parameter for more details. |
| birth_date | optional | string(20) | Birth date of the customer |
| customer_email | required | e-mail address | Must contain valid e-mail of customer |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

**Supported countries:**

| Country |
|---|
| CO |

Successful Response

```
stdClass Object
(
    [transaction_type] => baloto
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:21.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>baloto</transaction_type>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
  <timestamp>2020-07-15T12:08:21Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [transaction_type] => baloto
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 110
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:21.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>baloto</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>110</code>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:21Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

**BANCO DE OCCIDENTE**

Banco de Occidente is a cash payment method for Colombia

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\CashPayments\BancoDeOccidente');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setNationalId('8812128812')
        ->setBirthDate('30-12-1992')
        ->setCustomerEmail('travis@example.com')


        // Billing Address
        ->setBillingFirstName('Barney')
        ->setBillingLastName('Rubble')
        ->setBillingAddress1('14, Nerazdelni str')
        ->setBillingZipCode('1407')
        ->setBillingCity('Bogota')
        ->setBillingCountry('CO')


        // Risk Params
        ->setRiskUserId('123456');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```curl
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>banco_de_occidente</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <return_success_url>http://www.example.com/success</return_success_url>
    <return_failure_url>http://www.example.com/failure</return_failure_url>
    <amount>100</amount>
    <currency>USD</currency>
    <national_id>8812128812</national_id>
    <birth_date>30-12-1992</birth_date>
    <customer_email>travis@example.com</customer_email>
    <billing_address>
      <first_name>Barney</first_name>
      <last_name>Rubble</last_name>
      <address1>14, Nerazdelni str</address1>
      <zip_code>1407</zip_code>
      <city>Bogota</city>
      <country>CO</country>
    </billing_address>
    <risk_params>
      <user_id>123456</user_id>
    </risk_params>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **banco_de_occidente** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |

| Parameter | Required | Format | Description |
|---|---|---|---|
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| national_id | required | string(20) | National ID of the consumer. See Document ID Parameter for more details. |
| birth_date | optional | string(20) | Birth date of the customer |
| customer_email | required | e-mail address | Must contain valid e-mail of customer |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

**Supported countries:**

| Country |
|---|
| CO |

Successful Response

```
stdClass Object
(
    [transaction_type] => banco_de_occidente
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:21.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```


```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>banco_de_occidente</transaction_type>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
  <timestamp>2020-07-15T12:08:21Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```


**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response


```
stdClass Object
(
    [transaction_type] => banco_de_occidente
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 110
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:21.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```


```
This request is not implemented yet
```

```
This request is not implemented yet
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>banco_de_occidente</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>110</code>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:21Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

**BOLETO**

Boleto is a payment service in Brazil

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\CashPayments\Boleto');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setConsumerReference('barney_rubble')
        ->setNationalId('8812128812')
        ->setBirthDate('30-12-1992')
        ->setCustomerEmail('travis@example.com')


        // Billing Address
        ->setBillingFirstName('Barney')
        ->setBillingLastName('Rubble')
        ->setBillingAddress1('14, Nerazdelni str')
        ->setBillingZipCode('1407')
        ->setBillingCity('Rio de Janeiro')
        ->setBillingCountry('BR')


        // Risk Params
        ->setRiskUserId('123456');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```curl
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>boleto</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <return_success_url>http://www.example.com/success</return_success_url>
    <return_failure_url>http://www.example.com/failure</return_failure_url>
    <amount>100</amount>
    <currency>USD</currency>
    <consumer_reference>barney_rubble</consumer_reference>
    <national_id>8812128812</national_id>
    <birth_date>30-12-1992</birth_date>
    <customer_email>travis@example.com</customer_email>
    <billing_address>
      <first_name>Barney</first_name>
      <last_name>Rubble</last_name>
      <address1>14, Nerazdelni str</address1>
      <zip_code>1407</zip_code>
      <city>Rio de Janeiro</city>
      <country>BR</country>
    </billing_address>
    <risk_params>
      <user_id>123456</user_id>
    </risk_params>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **boleto** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |

| Parameter | Required | Format | Description |
|---|---|---|---|
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| consumer_reference | required | string(20) | Consumer reference is a unique consumer identifier |
| national_id | required | string(20) | National ID of the consumer. See Document ID Parameter for more details. |
| birth_date | optional | string(20) | Birth date of the customer |
| customer_email | required | e-mail address | Must contain valid e-mail of customer |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

**Supported countries:**

| Country |
|---|
| BR |

Successful Response

```
stdClass Object
(
    [transaction_type] => boleto
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:21.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>boleto</transaction_type>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
  <timestamp>2020-07-15T12:08:21Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [transaction_type] => boleto
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 110
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:21.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>boleto</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>110</code>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:21Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

**EFECTY**

Efecty is a cash-based payment method.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\CashPayments\Efecty');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setConsumerReference('barney_rumble')
        ->setNationalId('8812128812')
        ->setBirthDate('30-12-1992')
        ->setCustomerEmail('barney@example.com')


        // Billing Address
        ->setBillingFirstName('Barney')
        ->setBillingLastName('Rubble')
        ->setBillingAddress1('14, Nerazdelni str')
        ->setBillingZipCode('1407')
        ->setBillingCity('Bogota')
        ->setBillingCountry('CO')


        // Risk Params
        ->setRiskUserId('123456');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```curl
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>efecty</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <return_success_url>http://www.example.com/success</return_success_url>
    <return_failure_url>http://www.example.com/failure</return_failure_url>
    <amount>100</amount>
    <currency>USD</currency>
    <consumer_reference>barney_rumble</consumer_reference>
    <national_id>8812128812</national_id>
    <birth_date>30-12-1992</birth_date>
    <customer_email>barney@example.com</customer_email>
    <billing_address>
      <first_name>Barney</first_name>
      <last_name>Rubble</last_name>
      <address1>14, Nerazdelni str</address1>
      <zip_code>1407</zip_code>
      <city>Bogota</city>
      <country>CO</country>
    </billing_address>
    <risk_params>
      <user_id>123456</user_id>
    </risk_params>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **efecty** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |

| Parameter | Required | Format | Description |
|---|---|---|---|
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| consumer_reference | required | string(20) | Consumer reference is a unique consumer identifier |
| national_id | required | string(20) | National ID of the consumer. See Document ID Parameter for more details. |
| birth_date | optional | dd-mm-yyyy | Required for Visa only when MCC is a Financial Services one (e.g. MCC 6012) |
| customer_email | required | e-mail address | Must contain valid e-mail of customer |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

**Supported countries:**

| Country |
|---|
| CO |

Successful Response

```
stdClass Object
(
    [transaction_type] => efecty
    [status] => pending_async
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:21.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```


```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>efecty</transaction_type>
  <status>pending_async</status>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <timestamp>2020-07-15T12:08:21Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```


**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [transaction_type] => efecty
    [status] => error
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 110
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:21.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>efecty</transaction_type>
  <status>error</status>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>110</code>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:21Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

**OXXO**

OXXO is the preferred payment method in Mexico. It is a cash payment via a barcode document thats accepted in more than 14,000 stores.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\CashPayments\Oxxo');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setConsumerReference('barney_rumble')
        ->setNationalId('8812128812')
        ->setBirthDate('30-12-1992')
        ->setCustomerEmail('barney@example.com')


        // Billing Address
        ->setBillingFirstName('Barney')
        ->setBillingLastName('Rubble')
        ->setBillingAddress1('14, Nerazdelni str')
        ->setBillingZipCode('1407')
        ->setBillingCity('Mexico City')
        ->setBillingCountry('MX')


        // Risk Params
        ->setRiskUserId('123456');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```curl
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>oxxo</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <return_success_url>http://www.example.com/success</return_success_url>
    <return_failure_url>http://www.example.com/failure</return_failure_url>
    <amount>100</amount>
    <currency>USD</currency>
    <consumer_reference>barney_rumble</consumer_reference>
    <national_id>8812128812</national_id>
    <birth_date>30-12-1992</birth_date>
    <customer_email>barney@example.com</customer_email>
    <billing_address>
      <first_name>Barney</first_name>
      <last_name>Rubble</last_name>
      <address1>14, Nerazdelni str</address1>
      <zip_code>1407</zip_code>
      <city>Mexico City</city>
      <country>MX</country>
    </billing_address>
    <risk_params>
      <user_id>123456</user_id>
    </risk_params>
  </payment_transaction>'
```

### Request Parameters

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **oxxo** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |

| Parameter | Required | Format | Description |
|---|---|---|---|
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| consumer_reference | required | string(20) | Consumer reference is a unique consumer identifier |
| national_id | required | string(20) | National ID of the consumer. See Document ID Parameter for more details. |
| birth_date | optional | dd-mm-yyyy | Required for Visa only when MCC is a Financial Services one (e.g. MCC 6012) |
| customer_email | required | e-mail address | Must contain valid e-mail of customer |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

**Supported countries:**

| Country |
|---|
| MX |

Successful Response

```
stdClass Object
(
    [transaction_type] => oxxo
    [status] => pending_async
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:21.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>oxxo</transaction_type>
  <status>pending_async</status>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <timestamp>2020-07-15T12:08:21Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [transaction_type] => oxxo
    [status] => error
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 110
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:21.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>oxxo</transaction_type>
  <status>error</status>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>110</code>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:21Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

**PAGO FACIL**

Pago Facil is a cash-based payment used for online purchases.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\CashPayments\PagoFacil');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setConsumerReference('barney_rumble')
        ->setNationalId('8812128812')
        ->setBirthDate('30-12-1992')
        ->setCustomerEmail('barney@example.com')


        // Billing Address
        ->setBillingFirstName('Barney')
        ->setBillingLastName('Rubble')
        ->setBillingAddress1('14, Nerazdelni str')
        ->setBillingZipCode('1407')
        ->setBillingCity('Buenos Aires')
        ->setBillingCountry('AR')


        // Risk Params
        ->setRiskUserId('123456');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```curl
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>pago_facil</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <return_success_url>http://www.example.com/success</return_success_url>
    <return_failure_url>http://www.example.com/failure</return_failure_url>
    <amount>100</amount>
    <currency>USD</currency>
    <consumer_reference>barney_rumble</consumer_reference>
    <national_id>8812128812</national_id>
    <birth_date>30-12-1992</birth_date>
    <customer_email>barney@example.com</customer_email>
    <billing_address>
      <first_name>Barney</first_name>
      <last_name>Rubble</last_name>
      <address1>14, Nerazdelni str</address1>
      <zip_code>1407</zip_code>
      <city>Buenos Aires</city>
      <country>AR</country>
    </billing_address>
    <risk_params>
      <user_id>123456</user_id>
    </risk_params>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **pago_facil** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |

| Parameter | Required | Format | Description |
|---|---|---|---|
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| consumer_reference | required | string(20) | Consumer reference is a unique consumer identifier |
| national_id | required | string(20) | National ID of the consumer. See Document ID Parameter for more details. |
| birth_date | optional | dd-mm-yyyy | Required for Visa only when MCC is a Financial Services one (e.g. MCC 6012) |
| customer_email | required | e-mail address | Must contain valid e-mail of customer |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

**Supported countries:**

| Country |
|---|
| AR |

Successful Response

```
stdClass Object
(
    [transaction_type] => pago_facil
    [status] => pending_async
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:21.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```


```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>pago_facil</transaction_type>
  <status>pending_async</status>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <timestamp>2020-07-15T12:08:21Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```


**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [transaction_type] => pago_facil
    [status] => error
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 110
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:21.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>pago_facil</transaction_type>
  <status>error</status>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>110</code>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:21Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

**REDPAGOS**

Redpagos is a cash payment in Uruguay

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\CashPayments\Redpagos');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setConsumerReference('barney_rubble')
        ->setNationalId('8812128812')
        ->setBirthDate('30-12-1992')
        ->setCustomerEmail('travis@example.com')


        // Billing Address
        ->setBillingFirstName('Barney')
        ->setBillingLastName('Rubble')
        ->setBillingAddress1('14, Nerazdelni str')
        ->setBillingZipCode('1407')
        ->setBillingCity('Montevideo')
        ->setBillingCountry('UY')


        // Risk Params
        ->setRiskUserId('123456');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```curl
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>redpagos</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <return_success_url>http://www.example.com/success</return_success_url>
    <return_failure_url>http://www.example.com/failure</return_failure_url>
    <amount>100</amount>
    <currency>USD</currency>
    <consumer_reference>barney_rubble</consumer_reference>
    <national_id>8812128812</national_id>
    <birth_date>30-12-1992</birth_date>
    <customer_email>travis@example.com</customer_email>
    <billing_address>
      <first_name>Barney</first_name>
      <last_name>Rubble</last_name>
      <address1>14, Nerazdelni str</address1>
      <zip_code>1407</zip_code>
      <city>Montevideo</city>
      <country>UY</country>
    </billing_address>
    <risk_params>
      <user_id>123456</user_id>
    </risk_params>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **redpagos** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |

| Parameter | Required | Format | Description |
|---|---|---|---|
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| consumer_reference | required | string(20) | Consumer reference is a unique consumer identifier |
| national_id | required | string(20) | National ID of the consumer. See Document ID Parameter for more details. |
| birth_date | optional | string(20) | Birth date of the customer |
| customer_email | required | e-mail address | Must contain valid e-mail of customer |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

**Supported countries:**

| Country |
|---|
| UY |

Successful Response

```
stdClass Object
(
    [transaction_type] => redpagos
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:21.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>redpagos</transaction_type>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
  <timestamp>2020-07-15T12:08:21Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [transaction_type] => redpagos
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 110
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:21.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>redpagos</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>110</code>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:21Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

---

# Gift Cards

**INTERSOLVE**

**Intersolve transactions are made using gift card provided by Intersolve**

Using an intersolve transaction, the amount is immediately billed to the customer's gift card.

It can be reversed via a void transaction. Intersolve gift cards also supportpayout.

Use intersolve transactions if you are using gift cards provided by Intersolve.

This transaction type supports Tokenization.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\GiftCards\Intersolve');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardNumber('7000001163991388834')
        ->setCvv('944062')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.giftcards.IntersolveRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    IntersolveRequest request = new IntersolveRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardNumber("7000001163991388834");
    request.setCvv("944062");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
This request is not implemented yet
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>intersolve</transaction_type>
   <transaction_id>43671</transaction_id>
   <usage>40208 concert tickets</usage>
   <remote_ip>245.253.2.12</remote_ip>
   <amount>100</amount>
   <currency>USD</currency>
   <card_number>7000001163991388834</card_number>
   <cvv>944062</cvv>
   <billing_address>
     <first_name>Travis</first_name>
     <last_name>Pastrana</last_name>
     <address1>Muster Str. 12</address1>
     <zip_code>10178</zip_code>
     <city>Los Angeles</city>
     <state>CA</state>
     <country>US</country>
   </billing_address>
 </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **intersolve** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | required* | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| card_number | required | string(19..21) | Gift card number |
| cvv | required* | 5 to 8 digits | Verification code of the gift card, requirement is based on terminal configuration |
| token | optional | string(36) | See Tokenization for more details. If present, the cardholder parameters can be omitted. Cannot be set together with `remember_card` |
| remember_card | optional | "true" | See Tokenize. Tokenizes cardholder parameters. Cannot be set together with `token` |
| consumer_id | optional | string(10) | See Consumers and Tokenization. Combine with `remember_card` to tokenize or with `token` to use token |
| **billing_address** | required* | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required* | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |

| Parameter | Required | Format | Description |
|---|---|---|---|
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |
| **dynamic_descriptor_params** | optional | | |
| merchant_name | optional | string(25) | Allows to dynamically override the charge descriptor |
| merchant_city | optional | string(13) | Allows to dynamically override the merchant phone number |
| sub_merchant_id | optional | string(15) | Allows to dynamically override the sub-merchant ID. |

`required* = conditionally required`

Successful Response

```
stdClass Object
(
    [transaction_type] => intersolve
    [status] => approved
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [consumer_id] => 123456
    [token] => ee946db8-d7db-4bb7-b608-b65b153e127d
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:21.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[intersolve]>
    <status content=[approved]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <consumer_id content=[123456]>
    <token content=[ee946db8-d7db-4bb7-b608-b65b153e127d]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <timestamp content=[2020-07-15T12:08:21Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[true]>
]>
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>intersolve</transaction_type>
  <status>approved</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <consumer_id>123456</consumer_id>
  <token>ee946db8-d7db-4bb7-b608-b65b153e127d</token>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <timestamp>2020-07-15T12:08:21Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| transaction_id | string(255) | Unique transaction id defined by merchant |

| Parameter | Type | Description |
|---|---|---|
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| consumer_id | string(10) | Consumer unique reference. See Consumers |
| token | string(36) | Plain-text token value. See Tokenization |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08- 30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |
| dynamic_descriptor_params | section | Optional, returned only if dynamic desc params are submitted on the API. Note here that the formatted dyn desc params are returned - as they would be submitted to the schemes for settlement. |

Error Response

```
stdClass Object
(
    [transaction_type] => intersolve
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 340
    [technical_message] => Transaction_id is invalid!
    [message] => Transaction_id is invalid!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:21.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => false
)
```

```
<payment_response content=[
    <transaction_type content=[intersolve]>
    <status content=[error]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <code content=[340]>
    <technical_message content=[Transaction_id is invalid!]>
    <message content=[Transaction_id is invalid!]>
    <timestamp content=[2020-07-15T12:08:21Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[false]>
]>
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>intersolve</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>340</code>
  <technical_message>Transaction_id is invalid!</technical_message>
  <message>Transaction_id is invalid!</message>
  <timestamp>2020-07-15T12:08:21Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>false</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08- 30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |
| dynamic_descriptor_params | section | Optional, returned only if dynamic desc params are submitted on the API. Note here that the formatted dyn desc params are returned - as they would be submitted to the schemes for settlement. |

#### FASHIONCHEQUE

**Fashioncheque transactions are made using gift card provided by Fashioncheque**

Using a fashioncheque transaction, the amount is immediately billed to the customer's gift card.

It can be reversed via a void transaction on the same day of the transaction. They can also be refunded.

Use fashioncheque transactions, if you are using gift cards provided by Fashioncheque.

> This transaction type supports Tokenization.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\GiftCards\Fashioncheque');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardNumber('6046425117120757123')
        ->setCvv('121839')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.giftcards.FashionchequeRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    FashionchequeRequest request = new FashionchequeRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardNumber("6046425117120757123");
    request.setCvv("121839");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
This request is not implemented yet
```

```curl
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>fashioncheque</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>100</amount>
    <currency>USD</currency>
    <card_number>6046425117120757123</card_number>
    <cvv>121839</cvv>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>10178</zip_code>
      <city>Los Angeles</city>
      <state>CA</state>
      <country>US</country>
    </billing_address>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **fashioncheque** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | required* | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |

| Parameter | Required | Format | Description |
|---|---|---|---|
| amount | required | integer > 0 | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | required | string(3) | Only USD and EUR |
| card_number | required | string(19..21) | Gift card number |
| cvv | required* | 5 to 8 digits | Verification code of the gift card, requirement is based on terminal configuration |
| token | optional | string(36) | See Tokenization for more details. If present, the cardholder parameters can be omitted |
| **billing_address** | required* | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required* | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |
| **dynamic_descriptor_params** | optional | | |
| merchant_name | optional | string(25) | Allows to dynamically override the charge descriptor |
| merchant_city | optional | string(13) | Allows to dynamically override the merchant phone number |
| sub_merchant_id | optional | string(15) | Allows to dynamically override the sub-merchant ID. |

`required* = conditionally required`

Successful Response

```
stdClass Object
(
    [transaction_type] => fashioncheque
    [status] => approved
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:21.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[fashioncheque]>
    <status content=[approved]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <timestamp content=[2020-07-15T12:08:21Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[true]>
]>
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>fashioncheque</transaction_type>
  <status>approved</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <timestamp>2020-07-15T12:08:21Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08- 30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |
| dynamic_descriptor_params | section | Optional, returned only if dynamic desc params are submitted on the API. Note here that the formatted dyn desc params are returned - as they would be submitted to the schemes for settlement. |

Error Response

```
stdClass Object
(
    [transaction_type] => fashioncheque
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 340
    [technical_message] => Transaction_id is invalid!
    [message] => Transaction_id is invalid!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:21.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => false
)
```

```
<payment_response content=[
    <transaction_type content=[fashioncheque]>
    <status content=[error]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <code content=[340]>
    <technical_message content=[Transaction_id is invalid!]>
    <message content=[Transaction_id is invalid!]>
    <timestamp content=[2020-07-15T12:08:21Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[false]>
]>
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>fashioncheque</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>340</code>
  <technical_message>Transaction_id is invalid!</technical_message>
  <message>Transaction_id is invalid!</message>
  <timestamp>2020-07-15T12:08:21Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>false</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08- 30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |
| dynamic_descriptor_params | section | Optional, returned only if dynamic desc params are submitted on the API. Note here that the formatted dyn desc params are returned - as they would be submitted to the schemes for settlement. |

**TCS**

The container store **transactions are made using gift cards provided by TCS**.

The amount from a Container Store Transactions is immediately billed to the customer's gift card.

It can be reversed via a void transaction.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\GiftCards\Tcs');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardNumber('6046425117120757123')
        ->setCvv('121839')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.giftcards.TCSRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    TCSRequest request = new TCSRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardNumber("6046425117120757123");
    request.setCvv("121839");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
This request is not implemented yet
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>container_store</transaction_type>
   <transaction_id>43671</transaction_id>
   <usage>40208 concert tickets</usage>
   <remote_ip>245.253.2.12</remote_ip>
   <amount>100</amount>
   <currency>USD</currency>
   <card_number>6046425117120757123</card_number>
   <cvv>121839</cvv>
   <billing_address>
     <first_name>Travis</first_name>
     <last_name>Pastrana</last_name>
     <address1>Muster Str. 12</address1>
     <zip_code>10178</zip_code>
     <city>Los Angeles</city>
     <state>CA</state>
     <country>US</country>
   </billing_address>
 </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **container_store** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | required* | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | required | string(3) | Only USD and EUR |
| card_number | required | string(19..21) | Gift card number |
| cvv | required* | 5 to 8 digits | Verification code of the gift card, requirement is based on terminal configuration |
| token | optional | string(36) | See Tokenization for more details. If present, the cardholder parameters can be omitted |
| **billing_address** | required* | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required* | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |
| **dynamic_descriptor_params** | optional | | |
| merchant_name | optional | string(25) | Allows to dynamically override the charge descriptor |

| Parameter | Required | Format | Description |
|---|---|---|---|
| merchant_city | optional | string(13) | Allows to dynamically override the merchant phone number |
| sub_merchant_id | optional | string(15) | Allows to dynamically override the sub-merchant ID. |

`required* = conditionally required`

Successful Response

```
stdClass Object
(
    [transaction_type] => container_store
    [status] => approved
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:21.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[container_store]>
    <status content=[approved]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <timestamp content=[2020-07-15T12:08:21Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[true]>
]>
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>container_store</transaction_type>
  <status>approved</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <timestamp>2020-07-15T12:08:21Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08- 30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |

| Parameter | Type | Description |
|---|---|---|
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |
| dynamic_descriptor_params | section | Optional, returned only if dynamic desc params are submitted on the API. Note here that the formatted dyn desc params are returned - as they would be submitted to the schemes for settlement. |

Error Response

```
stdClass Object
(
    [transaction_type] => container_store
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 340
    [technical_message] => Transaction_id is invalid!
    [message] => Transaction_id is invalid!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:21.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => false
)
```

```
<payment_response content=[
    <transaction_type content=[container_store]>
    <status content=[error]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <code content=[340]>
    <technical_message content=[Transaction_id is invalid!]>
    <message content=[Transaction_id is invalid!]>
    <timestamp content=[2020-07-15T12:08:21Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[false]>
]>
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>container_store</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>340</code>
  <technical_message>Transaction_id is invalid!</technical_message>
  <message>Transaction_id is invalid!</message>
  <timestamp>2020-07-15T12:08:21Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>false</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |

| Parameter | Type | Description |
|---|---|---|
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08- 30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |
| dynamic_descriptor_params | section | Optional, returned only if dynamic desc params are submitted on the API. Note here that the formatted dyn desc params are returned - as they would be submitted to the schemes for settlement. |

#### SPLIT PAYMENTS

Split payments are performed on gift card transaction types when there isn't enough balance in the gift card. In order to use split payments you need to enable them on terminal. For more information please contact tech support.

**Split payments follow this workflow:**

- Split payment can be initiated only on gift card transaction.
- You can have maximum three payment series including the initial transaction.
- If the gift card does not have enough balance to perform the transaction, the whole available balance is taken from the gift card and new split payment is initiated.
- You can continue the split payment with another gift card
- You can finish the split payment with either gift card or credit card by submitting the 'unique id' of the initial transaction as 'reference id' in the request.
- Any failure during split payment causes rollback of all split payment series transactions.

> Credit card transaction can only be last in split payment series and any series transactions must be submitted with the actual leftover amount.

**Example for initial split payment transaction:**

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\GiftCards\Fashioncheque');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('5000')
        ->setCurrency('EUR')
        ->setCardNumber('6046425117120757123')
        ->setCvv('121839')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.giftcards.FashionchequeRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    FashionchequeRequest request = new FashionchequeRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(5000));
    request.setCurrency("EUR");
    request.setCardNumber("6046425117120757123");
    request.setCvv("121839");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
This request is not implemented yet
```

```curl
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>fashioncheque</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>5000</amount>
    <currency>EUR</currency>
    <card_number>6046425117120757123</card_number>
    <cvv>121839</cvv>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>10178</zip_code>
      <city>Los Angeles</city>
      <state>CA</state>
      <country>US</country>
    </billing_address>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **fashioncheque** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | required* | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |

| Parameter | Required | Format | Description |
|---|---|---|---|
| amount | required | integer > 0 | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | required | string(3) | Only USD and EUR |
| card_number | required | string(19..21) | Gift card number |
| cvv | required* | 5 to 8 digits | Verification code of the gift card, requirement is based on terminal configuration |
| token | optional | string(36) | See Tokenization for more details. If present, the cardholder parameters can be omitted |
| **billing_address** | required* | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required* | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |
| **dynamic_descriptor_params** | optional | | |
| merchant_name | optional | string(25) | Allows to dynamically override the charge descriptor |
| merchant_city | optional | string(13) | Allows to dynamically override the merchant phone number |
| sub_merchant_id | optional | string(15) | Allows to dynamically override the sub-merchant ID. |

`required* = conditionally required`

Successful Response

```
stdClass Object
(
    [transaction_type] => fashioncheque
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 406bc1b340472db4dbbba4b749850234
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:21.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 2000
    [currency] => EUR
    [split_payment] => initiated
    [leftover_amount] => 3000
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[fashioncheque]>
    <status content=[pending_async]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[406bc1b340472db4dbbba4b749850234]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <timestamp content=[2020-07-15T12:08:21Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[2000]>
    <currency content=[EUR]>
    <split_payment content=[initiated]>
    <leftover_amount content=[3000]>
    <sent_to_acquirer content=[true]>
]>
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>fashioncheque</transaction_type>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>406bc1b340472db4dbbba4b749850234</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <timestamp>2020-07-15T12:08:21Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>2000</amount>
  <currency>EUR</currency>
  <split_payment>initiated</split_payment>
  <leftover_amount>3000</leftover_amount>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08- 30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| split_payment | string | Split payment status for this transaction. Should be 'initiated'. |
| leftover_amount | integer | Leftover amount of transaction in minor currency unit, see Currency Handling for details |
| sent_to_acquirer | string(255) | "true" or "false" |
| dynamic_descriptor_params | section | Optional, returned only if dynamic desc params are submitted on the API. Note here that the formatted dyn desc params are returned - as they would be submitted to the schemes for settlement. |

Error Response

```
stdClass Object
(
    [transaction_type] => fashioncheque
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 340
    [technical_message] => Transaction_id is invalid!
    [message] => Transaction_id is invalid!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:21.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 5000
    [currency] => EUR
    [sent_to_acquirer] => false
)
```

```
<payment_response content=[
    <transaction_type content=[fashioncheque]>
    <status content=[error]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <code content=[340]>
    <technical_message content=[Transaction_id is invalid!]>
    <message content=[Transaction_id is invalid!]>
    <timestamp content=[2020-07-15T12:08:21Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[5000]>
    <currency content=[EUR]>
    <sent_to_acquirer content=[false]>
]>
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>fashioncheque</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>340</code>
  <technical_message>Transaction_id is invalid!</technical_message>
  <message>Transaction_id is invalid!</message>
  <timestamp>2020-07-15T12:08:21Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>5000</amount>
  <currency>EUR</currency>
  <sent_to_acquirer>false</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08- 30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |
| dynamic_descriptor_params | section | Optional, returned only if dynamic desc params are submitted on the API. Note here that the formatted dyn desc params are returned - as they would be submitted to the schemes for settlement. |

**Example for continued split payment transaction:**

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\GiftCards\Intersolve');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setReferenceId('406bc1b340472db4dbbba4b749850234')
        ->setAmount('3000')
        ->setCurrency('EUR')
        ->setCardNumber('7000001163991388834')
        ->setCvv('944062')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.giftcards.IntersolveRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    IntersolveRequest request = new IntersolveRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setReferenceId("406bc1b340472db4dbbba4b749850234");
    request.setAmount(new BigDecimal(3000));
    request.setCurrency("EUR");
    request.setCardNumber("7000001163991388834");
    request.setCvv("944062");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
This request is not implemented yet
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>intersolve</transaction_type>
   <transaction_id>43671</transaction_id>
   <usage>40208 concert tickets</usage>
   <remote_ip>245.253.2.12</remote_ip>
   <reference_id>406bc1b340472db4dbbba4b749850234</reference_id>
   <amount>3000</amount>
   <currency>EUR</currency>
   <card_number>7000001163991388834</card_number>
   <cvv>944062</cvv>
   <billing_address>
     <first_name>Travis</first_name>
     <last_name>Pastrana</last_name>
     <address1>Muster Str. 12</address1>
     <zip_code>10178</zip_code>
     <city>Los Angeles</city>
     <state>CA</state>
     <country>US</country>
   </billing_address>
 </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **intersolve** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | required* | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| reference_id | required | string(32) | The 'unique id' of the initial split payment transaction. |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| card_number | required | string(19..21) | Gift card number |
| cvv | required* | 5 to 8 digits | Verification code of the gift card, requirement is based on terminal configuration |
| token | optional | string(36) | See Tokenization for more details. If present, the cardholder parameters can be omitted |
| **billing_address** | required* | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required* | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |

| Parameter | Required | Format | Description |
|---|---|---|---|
| country | optional | string(2) | Country code in ISO 3166 |
| **dynamic_descriptor_params** | optional | | |
| merchant_name | optional | string(25) | Allows to dynamically override the charge descriptor |
| merchant_city | optional | string(13) | Allows to dynamically override the merchant phone number |
| sub_merchant_id | optional | string(15) | Allows to dynamically override the sub-merchant ID. |

`required* = conditionally required`

Successful Response

```
stdClass Object
(
    [transaction_type] => intersolve
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:21.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 2500
    [currency] => EUR
    [split_payment] => continued
    [leftover_amount] => 500
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[intersolve]>
    <status content=[pending_async]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <timestamp content=[2020-07-15T12:08:21Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[2500]>
    <currency content=[EUR]>
    <split_payment content=[continued]>
    <leftover_amount content=[500]>
    <sent_to_acquirer content=[true]>
]>
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>intersolve</transaction_type>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <timestamp>2020-07-15T12:08:21Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>2500</amount>
  <currency>EUR</currency>
  <split_payment>continued</split_payment>
  <leftover_amount>500</leftover_amount>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| transaction_id | string(255) | Unique transaction id defined by merchant |

| Parameter | Type | Description |
|---|---|---|
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08- 30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| split_payment | string | Split payment status for this transaction. Should be 'continued'. |
| leftover_amount | integer | Leftover amount of transaction in minor currency unit, see Currency Handling for details |
| sent_to_acquirer | string(255) | "true" or "false" |
| dynamic_descriptor_params | section | Optional, returned only if dynamic desc params are submitted on the API. Note here that the formatted dyn desc params are returned - as they would be submitted to the schemes for settlement. |

Error Response

```
stdClass Object
(
    [transaction_type] => intersolve
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 340
    [technical_message] => Transaction_id is invalid!
    [message] => Transaction_id is invalid!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:21.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 3000
    [currency] => EUR
    [sent_to_acquirer] => false
)
```

```
<payment_response content=[
    <transaction_type content=[intersolve]>
    <status content=[error]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <code content=[340]>
    <technical_message content=[Transaction_id is invalid!]>
    <message content=[Transaction_id is invalid!]>
    <timestamp content=[2020-07-15T12:08:21Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[3000]>
    <currency content=[EUR]>
    <sent_to_acquirer content=[false]>
]>
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>intersolve</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>340</code>
  <technical_message>Transaction_id is invalid!</technical_message>
  <message>Transaction_id is invalid!</message>
  <timestamp>2020-07-15T12:08:21Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>3000</amount>
  <currency>EUR</currency>
  <sent_to_acquirer>false</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08- 30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |
| dynamic_descriptor_params | section | Optional, returned only if dynamic desc params are submitted on the API. Note here that the formatted dyn desc params are returned - as they would be submitted to the schemes for settlement. |

**Example for finalized split payment transaction:**

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\GiftCards\Intersolve');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setReferenceId('406bc1b340472db4dbbba4b749850234')
        ->setAmount('500')
        ->setCurrency('EUR')
        ->setCardNumber('7000001163991388834')
        ->setCvv('944062')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.giftcards.IntersolveRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    IntersolveRequest request = new IntersolveRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setReferenceId("406bc1b340472db4dbbba4b749850234");
    request.setAmount(new BigDecimal(500));
    request.setCurrency("EUR");
    request.setCardNumber("7000001163991388834");
    request.setCvv("944062");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
This request is not implemented yet
```

```xml
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>intersolve</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <reference_id>406bc1b340472db4dbbba4b749850234</reference_id>
    <amount>500</amount>
    <currency>EUR</currency>
    <card_number>7000001163991388834</card_number>
    <cvv>944062</cvv>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>10178</zip_code>
      <city>Los Angeles</city>
      <state>CA</state>
      <country>US</country>
    </billing_address>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **intersolve** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | required* | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |

| Parameter | Required | Format | Description |
|---|---|---|---|
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| reference_id | required | string(32) | The 'unique id' of the initial split payment transaction. |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| card_number | required | string(19..21) | Gift card number |
| cvv | required* | 5 to 8 digits | Verification code of the gift card, requirement is based on terminal configuration |
| token | optional | string(36) | See Tokenization for more details. If present, the cardholder parameters can be omitted |
| **billing_address** | required* | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required* | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |
| **dynamic_descriptor_params** | optional | | |
| merchant_name | optional | string(25) | Allows to dynamically override the charge descriptor |
| merchant_city | optional | string(13) | Allows to dynamically override the merchant phone number |
| sub_merchant_id | optional | string(15) | Allows to dynamically override the sub-merchant ID. |

`required* = conditionally required`

Successful Response

```
stdClass Object
(
    [transaction_type] => intersolve
    [status] => approved
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:22.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 500
    [currency] => EUR
    [split_payment] => finalized
    [leftover_amount] => 0
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[intersolve]>
    <status content=[approved]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <timestamp content=[2020-07-15T12:08:22Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[500]>
    <currency content=[EUR]>
    <split_payment content=[finalized]>
    <leftover_amount content=[0]>
    <sent_to_acquirer content=[true]>
]>
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>intersolve</transaction_type>
  <status>approved</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <timestamp>2020-07-15T12:08:22Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>500</amount>
  <currency>EUR</currency>
  <split_payment>finalized</split_payment>
  <leftover_amount>0</leftover_amount>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08- 30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| split_payment | string | Split payment status for this transaction. Should be 'finalized'. |
| leftover_amount | integer | Leftover amount of transaction in minor currency unit, seeCurrency Handling for details |
| sent_to_acquirer | string(255) | "true" or "false" |
| dynamic_descriptor_params | section | Optional, returned only if dynamic desc params are submitted on the API. Note here that the formatted dyn desc params are returned - as they would be submitted to the schemes for settlement. |

Error Response

```
stdClass Object
(
    [transaction_type] => intersolve
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 340
    [technical_message] => Transaction_id is invalid!
    [message] => Transaction_id is invalid!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:22.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 500
    [currency] => EUR
    [sent_to_acquirer] => false
)
```

```
<payment_response content=[
    <transaction_type content=[intersolve]>
    <status content=[error]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <code content=[340]>
    <technical_message content=[Transaction_id is invalid!]>
    <message content=[Transaction_id is invalid!]>
    <timestamp content=[2020-07-15T12:08:22Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[500]>
    <currency content=[EUR]>
    <sent_to_acquirer content=[false]>
]>
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>intersolve</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>340</code>
  <technical_message>Transaction_id is invalid!</technical_message>
  <message>Transaction_id is invalid!</message>
  <timestamp>2020-07-15T12:08:22Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>500</amount>
  <currency>EUR</currency>
  <sent_to_acquirer>false</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08- 30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |
| dynamic_descriptor_params | section | Optional, returned only if dynamic desc params are submitted on the API. Note here that the formatted dyn desc params are returned - as they would be submitted to the schemes for settlement. |

Split Payment can also be finalized using Sale or Sale3D

**Example for finalized Sale split payment transaction:**

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Sale');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setReferenceId('406bc1b340472db4dbbba4b749850234')
        ->setAmount('500')
        ->setCurrency('EUR')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('4200000000000000')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.SaleRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    SaleRequest request = new SaleRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setReferenceId("406bc1b340472db4dbbba4b749850234");
    request.setAmount(new BigDecimal(500));
    request.setCurrency("EUR");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("4200000000000000");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.sale(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "reference_id": "406bc1b340472db4dbbba4b749850234",
    "amount": "500",
    "currency": "EUR",
    "card_holder": "Travis Pastrana",
    "card_number": "4200000000000000",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>sale</transaction_type>
   <transaction_id>43671</transaction_id>
   <usage>40208 concert tickets</usage>
   <remote_ip>245.253.2.12</remote_ip>
   <reference_id>406bc1b340472db4dbbba4b749850234</reference_id>
   <amount>500</amount>
   <currency>EUR</currency>
   <card_holder>Travis Pastrana</card_holder>
   <card_number>4200000000000000</card_number>
   <expiration_month>12</expiration_month>
   <expiration_year>2021</expiration_year>
   <cvv>834</cvv>
   <customer_email>travis@example.com</customer_email>
   <customer_phone>+1987987987987</customer_phone>
   <billing_address>
     <first_name>Travis</first_name>
     <last_name>Pastrana</last_name>
     <address1>Muster Str. 12</address1>
     <zip_code>10178</zip_code>
     <city>Los Angeles</city>
     <state>CA</state>
     <country>US</country>
   </billing_address>
 </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **sale** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| gaming | optional | "true" | Signifies whether a gaming transaction is performed. Gaming transactions usually use MCC 7995. Contacttech-support@e-comprocessing.com for more details |
| moto | optional | "true" | Signifies whether a MOTO (mail order telephone order) transaction is performed. Contacttech-support@e-comprocessing.com for more details |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| reference_id | required* | string(32) | The 'unique id' of the initial split payment transaction |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| card_holder | required | string(255) | Full name of customer as printed on credit card (first name and last name at least) |
| card_number | required | 13 to 16 digits | Complete cc number of customer |
| cvv | required* | 3 to 4 digits | cvv of cc, requirement is based on terminal configuration |
| expiration_month | required | MM | Expiration month as printed on credit card |
| expiration_year | required | YYYY | Expiration year as printed on credit card |
| customer_email | required* | e-mail address | Must contain valid e-mail of customer |
| customer_phone | required* | string(32) | Must contain valid phone number of customer |
| birth_date | required* | dd-mm-yyyy | Required for Visa only when MCC is a Financial Services one (e.g. MCC 6012) |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |

| Parameter | Required | Format | Description |
|---|---|---|---|
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required* | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |
| **dynamic_descriptor_params** | optional | | |
| merchant_name | optional | string(25) | Allows to dynamically override the charge descriptor |
| merchant_city | optional | string(13) | Allows to dynamically override the merchant phone number |
| sub_merchant_id | optional | string(15) | Allows to dynamically override the sub-merchant ID. |

`required* = conditionally required`

Successful Response

```
stdClass Object
(
    [transaction_type] => sale
    [status] => approved
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [avs_response_code] => 5I
    [avs_response_text] => Response provided by issuer processor; Address information not verified
    [authorization_code] => 345678
    [response_code] => 00
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:22.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 500
    [currency] => EUR
    [split_payment] => finalized
    [leftover_amount] => 0
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[sale]>
    <status content=[approved]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <avs_response_code content=[5I]>
    <avs_response_text content=[Response provided by issuer processor; Address information not verified]>
    <authorization_code content=[345678]>
    <response_code content=[00]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <timestamp content=[2020-07-15T12:08:22Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[500]>
    <currency content=[EUR]>
    <split_payment content=[finalized]>
    <leftover_amount content=[0]>
    <sent_to_acquirer content=[true]>
]>
```

```
{
    transaction_type: "sale",
    status: "approved",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    avs_response_code: "5I",
    avs_response_text: "Response provided by issuer processor; Address information not verified",
    authorization_code: "345678",
    response_code: "00",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    timestamp: "2020-07-15T12:08:22Z",
    descriptor: "Descriptor one",
    amount: "500",
    currency: "EUR",
    split_payment: "finalized",
    leftover_amount: "0",
    sent_to_acquirer: "true",
}
```



```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>sale</transaction_type>
  <status>approved</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <avs_response_code>5I</avs_response_code>
  <avs_response_text>Response provided by issuer processor; Address information not verified</avs_response_text>
  <authorization_code>345678</authorization_code>
  <response_code>00</response_code>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <timestamp>2020-07-15T12:08:22Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>500</amount>
  <currency>EUR</currency>
  <split_payment>finalized</split_payment>
  <leftover_amount>0</leftover_amount>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```


**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| gaming | string(255) | Signifies whether a gaming transaction is performed. Gaming transactions usually use MCC 7995, contact tech support for more details. |
| moto | string(255) | Signifies whether a MOTO (mail order telephone order) transaction is performed. Contact tech support for more details. |
| avs_response_code | string(255) | Generated by the card network on trying to match the billing address when performing the address verification. Optional, returned if config is enabled and acquirer supports it. Check AVS Status Codes for details. |
| avs_response_text | string(255) | Gives the human response text for the AVS response code above. Optional, returned if config is enabled and acquirer supports it. Check AVS Status Codes for details. |
| authorization_code | string(6) | Generated by the card network when an authorisation has occurred, used to identify that auth. Consists of 6 alphanumeric chars |
| response_code | string(2) | Defines the issuer result of a transaction, the status of a message or some action taken or required. See issuer response codes for details |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08- 30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| split_payment | string | Split payment status for this transaction. Should be 'finalized'. |
| leftover_amount | integer | Leftover amount of transaction in minor currency unit, see Currency Handling for details |
| sent_to_acquirer | string(255) | "true" or "false" |

| Parameter | Type | Description |
|---|---|---|
| partial_approval | string(4) | Optional, set to 'true' if partial approval happened. Partially approved amount is then in the amount field. Check Partial Approvals for details |
| dynamic_descriptor_params | section | Optional, returned only if dynamic desc params are submitted on the API. Note here that the formatted dyn desc params are returned - as they would be submitted to the schemes for settlement. |

Error Response

```
stdClass Object
(
    [response_code] => 57
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 340
    [technical_message] => billing_address[zip_code] is invalid!
    [message] => billing_address[zip_code] is invalid!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:22.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 500
    [currency] => EUR
    [sent_to_acquirer] => false
)
```

```
<payment_response content=[
    <response_code content=[57]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <code content=[340]>
    <technical_message content=[billing_address[zip_code] is invalid!]>
    <message content=[billing_address[zip_code] is invalid!]>
    <timestamp content=[2020-07-15T12:08:22Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[500]>
    <currency content=[EUR]>
    <sent_to_acquirer content=[false]>
]>
```

```
{
    response_code: "57",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    code: "340",
    technical_message: "billing_address[zip_code] is invalid!",
    message: "billing_address[zip_code] is invalid!",
    timestamp: "2020-07-15T12:08:22Z",
    descriptor: "Descriptor one",
    amount: "500",
    currency: "EUR",
    sent_to_acquirer: "false",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <response_code>57</response_code>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>340</code>
  <technical_message>billing_address[zip_code] is invalid!</technical_message>
  <message>billing_address[zip_code] is invalid!</message>
  <timestamp>2020-07-15T12:08:22Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>500</amount>
  <currency>EUR</currency>
  <sent_to_acquirer>false</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| gaming | string(255) | Signifies whether a gaming transaction is performed. Gaming transactions usually use MCC 7995, contact tech support for more details. |
| moto | string(255) | Signifies whether a MOTO (mail order telephone order) transaction is performed. Contact tech support for more details. |
| response_code | string(2) | Defines the issuer result of a transaction, the status of a message or some action taken or required. See issuer response codes for details |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08- 30T17:46:11Z |

| Parameter | Type | Description |
|---|---|---|
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |
| dynamic_descriptor_params | section | Optional, returned only if dynamic desc params are submitted on the API. Note here that the formatted dyn desc params are returned - as they would be submitted to the schemes for settlement. |

**Example for finalized Sale 3D split payment transaction:**

Asynchronous 3 D Sv1 Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Sale3D');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setReferenceId('406bc1b340472db4dbbba4b749850234')
        ->setAmount('500')
        ->setCurrency('EUR')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('4711100000000000')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')


        // MPI Asynchronous
        ->setNotificationUrl('https://www.example.com/notification')
        ->setReturnSuccessUrl('http://www.example.com/success.html')
        ->setReturnFailureUrl('http://www.example.com/failure.html');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.Sale3DRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    Sale3DRequest request = new Sale3DRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setReferenceId("406bc1b340472db4dbbba4b749850234");
    request.setAmount(new BigDecimal(500));
    request.setCurrency("EUR");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("4711100000000000");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");


    // Mpi Asynchronous
    request.setNotificationUrl("https://www.example.com/notification");
    request.setReturnSuccessUrl("http://www.example.com/success.html");
    request.setReturnFailureUrl("http://www.example.com/failure.html");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.sale3d(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "reference_id": "406bc1b340472db4dbbba4b749850234",
    "amount": "500",
    "currency": "EUR",
    "card_holder": "Travis Pastrana",
    "card_number": "4711100000000000",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    },
    "notification_url": "https://www.example.com/notification",
    "return_success_url": "http://www.example.com/success.html",
    "return_failure_url": "http://www.example.com/failure.html"
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>sale3d</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <reference_id>406bc1b340472db4dbbba4b749850234</reference_id>
    <amount>500</amount>
    <currency>EUR</currency>
    <card_holder>Travis Pastrana</card_holder>
    <card_number>4711100000000000</card_number>
    <expiration_month>12</expiration_month>
    <expiration_year>2021</expiration_year>
    <cvv>834</cvv>
    <customer_email>travis@example.com</customer_email>
    <customer_phone>+1987987987987</customer_phone>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>10178</zip_code>
      <city>Los Angeles</city>
      <state>CA</state>
      <country>US</country>
    </billing_address>
    <payment_transaction>mpi_asynchronous</payment_transaction>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **sale3d** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| gaming | optional | "true" | Signifies whether a gaming transaction is performed. Gaming transactions usually use MCC 7995. Contacttech-support@e-comprocessing.com for more details |
| moto | optional | "true" | Signifies whether a MOTO (mail order telephone order) transaction is performed. Contacttech-support@e-comprocessing.com for more details |

| Parameter | Required | Format | Description |
|---|---|---|---|
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| reference_id | required | string(32) | The 'unique id' of the initial split payment transaction |
| notification_url | required[1] | url | URL at merchant where gateway sends outcome of transaction. |
| return_success_url | required[1] | url | URL where customer is sent to after successful payment |
| return_failure_url | required[1] | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| card_holder | required | string(255) | Full name of customer as printed on credit card (first name and last name at least) |
| card_number | required | 13 to 16 digits | Complete cc number of customer |
| cvv | required | 3 to 4 digits | cvv of cc, requirement is based on terminal configuration |
| expiration_month | required | MM | Expiration month as printed on credit card |
| expiration_year | required | YYYY | Expiration year as printed on credit card |
| customer_email | required* | e-mail address | Must contain valid e-mail of customer |
| customer_phone | required* | string(32) | Must contain valid phone number of customer |
| birth_date | required* | dd-mm-yyyy | Required for Visa only when MCC is a Financial Services one (e.g. MCC 6012) |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required* | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |
| **mpi_params** | required[2] | | |
| cavv | required[3] | string(255) | Verification Id of the authentication. Please note this can be the CAVV for Visa Card or UCAF to identify MasterCard. |
| eci | required[3] | string(255) | See Electronic Commerce Indicator as returned from the MPI for details |
| xid | required[3] | string(255) | Transaction ID generated by the 3D Secure service that uniquely identifies a 3D Secure check request |
| **dynamic_descriptor_params** | optional | | |
| merchant_name | optional | string(25) | Allows to dynamically override the charge descriptor |
| merchant_city | optional | string(13) | Allows to dynamically override the merchant phone number |
| sub_merchant_id | optional | string(15) | Allows to dynamically override the sub-merchant ID. |

`required* = conditionally required`

`1 - required if mpi_params is not present, the transaction will be handled asynchronously. Not required if configured on Terminal or Merchant level. Contact [tech-support@#{email_domain_name}](mailto:tech-support@#{email_domain_name}) for more details.`

`2 - required if transaction should be handled synchronous.`

`3 - eci is always required if mpi_params is present. xid and cavv are not required for the 3D attempted only workflow, where the merchant does not have the xid/cavv, only the eci.`

Successful Asynchronous Response

```
stdClass Object
(
    [transaction_type] => sale3d
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:22.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 500
    [currency] => EUR
    [split_payment] => finalized
    [leftover_amount] => 0
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[sale3d]>
    <status content=[pending_async]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <redirect_url content=[https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61]>
    <timestamp content=[2020-07-15T12:08:22Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[500]>
    <currency content=[EUR]>
    <split_payment content=[finalized]>
    <leftover_amount content=[0]>
    <sent_to_acquirer content=[true]>
]>
```

```
{
    transaction_type: "sale3d",
    status: "pending_async",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    redirect_url: "https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61",
    timestamp: "2020-07-15T12:08:22Z",
    descriptor: "Descriptor one",
    amount: "500",
    currency: "EUR",
    split_payment: "finalized",
    leftover_amount: "0",
    sent_to_acquirer: "true",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>sale3d</transaction_type>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
  <timestamp>2020-07-15T12:08:22Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>500</amount>
  <currency>EUR</currency>
  <split_payment>finalized</split_payment>
  <leftover_amount>0</leftover_amount>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| gaming | string(255) | Signifies whether a gaming transaction is performed. Gaming transactions usually use MCC 7995, contact tech support for more details. |
| moto | string(255) | Signifies whether a MOTO (mail order telephone order) transaction is performed. Contact tech support for more details. |
| avs_response_code | string(255) | Generated by the card network on trying to match the billing address when performing the address verification. Optional, returned if config is enabled and acquirer supports it. Check AVS Status Codes for details. |
| avs_response_text | string(255) | Gives the human response text for the AVS response code above. Optional, returned if config is enabled and acquirer supports it. Check AVS Status Codes for details. |
| authorization_code | string(6) | Generated by the card network when an authorisation has occurred, used to identify that auth. Consists of 6 alphanumeric chars |
| response_code | string(2) | Defines the issuer result of a transaction, the status of a message or some action taken or required. See issuer response codes for details |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08- 30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| partial_approval | string(4) | Optional, set to 'true' if partial approval happened. Partially approved amount is then in the amount field. Check Partial Approvals for details |
| split_payment | string | Split payment status for this transaction. Should be 'finalized'. |
| leftover_amount | integer | Leftover amount of transaction in minor currency unit, see Currency Handling for details |
| sent_to_acquirer | string(255) | "true" or "false" |
| dynamic_descriptor_params | section | Optional, returned only if dynamic desc params are submitted on the API. Note here that the formatted dyn desc params are returned - as they would be submitted to the schemes for settlement. |

Error Response

```
stdClass Object
(
    [transaction_type] => sale3d
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [response_code] => 57
    [code] => 340
    [technical_message] => expiration_year is invalid
    [message] => expiration_year is invalid
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:22.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 500
    [currency] => EUR
    [sent_to_acquirer] => false
)
```

```
<payment_response content=[
    <transaction_type content=[sale3d]>
    <status content=[error]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <response_code content=[57]>
    <code content=[340]>
    <technical_message content=[expiration_year is invalid]>
    <message content=[expiration_year is invalid]>
    <timestamp content=[2020-07-15T12:08:22Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[500]>
    <currency content=[EUR]>
    <sent_to_acquirer content=[false]>
]>
```

```
{
    transaction_type: "sale3d",
    status: "error",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    response_code: "57",
    code: "340",
    technical_message: "expiration_year is invalid",
    message: "expiration_year is invalid",
    timestamp: "2020-07-15T12:08:22Z",
    descriptor: "Descriptor one",
    amount: "500",
    currency: "EUR",
    sent_to_acquirer: "false",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>sale3d</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <response_code>57</response_code>
  <code>340</code>
  <technical_message>expiration_year is invalid</technical_message>
  <message>expiration_year is invalid</message>
  <timestamp>2020-07-15T12:08:22Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>500</amount>
  <currency>EUR</currency>
  <sent_to_acquirer>false</sent_to_acquirer>
</payment_transaction>
```


**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| gaming | string(255) | Signifies whether a gaming transaction is performed. Gaming transactions usually use MCC 7995, contact tech support for more details. |
| moto | string(255) | Signifies whether a MOTO (mail order telephone order) transaction is performed. Contact tech support for more details. |
| response_code | string(2) | Defines the issuer result of a transaction, the status of a message or some action taken or required. See issuer response codes for details |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08- 30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |
| dynamic_descriptor_params | section | Optional, returned only if dynamic desc params are submitted on the API. Note here that the formatted dyn desc params are returned - as they would be submitted to the schemes for settlement. |

**Voiding of split payments:**

- Voiding of any transaction in unfinished split payment (while still pending async) will cause rollback of all other transactions in split payment series. The current transaction will appear as voided while the other transaction will become declined.
- Voiding any transaction after the split payment has been completed will cause voiding only of the single transaction. In order to revert such split payment you need to manually void all transactions in it.
- The above points are valid also for refunding the transactions if the gift card gateway supports refund.

**Split Payment Timeouts:**

- All unfinished split payments will be automatically timed out after a period of time and all transactions in them will be rollbacked.
- If the split payment is finished with async 3D credit card transaction the split payment will be timed out according to the mpi timeout of the final transaction.

# Invoice Payment Methods

Alternative payments refer to payment methods that are used as an alternative to credit card payments.

Each alternative payment method has its own unique application, settlement process and currency support.

#### KLARNA AUTHORIZE

> Klarna is a Swedish e-commerce company that provides payment services for online stores.

**With Klarna Authorize transactions, you can confirm that an order is successful.**

After settling the transaction (e.g. shipping the goods), you should use klarna capture transaction type to capture the amount.

Klarna authorize transaction will automatically be cancelled after a certain time frame, most likely two weeks.

For a typical e-commerce application it is recommended to authorize the amount on incoming orders and capture it when shipping the goods.

If you choose not to serve the customer, consider to void the klarna authorize to cancel the initial transaction.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Alternatives\Klarna\Authorize');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setNotificationUrl('https://www.example.com/notification')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setReturnCancelUrl('http://www.example.com/cancel')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('60')
        ->setCurrency('USD')
        ->setCustomerPhone('+1987987987987')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerGender('male')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')


        // Shipping Address
        ->setShippingFirstName('Travis')
        ->setShippingLastName('Pastrana')
        ->setShippingAddress1('Muster Str. 12')
        ->setShippingZipCode('10001')
        ->setShippingCity('Los Angeles')
        ->setShippingState('CA')
        ->setShippingCountry('US');

    // Klarna Items
    $items = new \Genesis\API\Request\Financial\Alternatives\Klarna\Items('USD');
    $item  = new \Genesis\API\Request\Financial\Alternatives\Klarna\Item();
    $item
        ->setItemType('physical')
        ->setReference('19-402-USA')
        ->setName('BatteryPowerPack')
        ->setQuantity('1')
        ->setUnitPrice('60')
        ->setTaxRate('0')
        ->setTotalAmount('60')
        ->setTotalDiscountAmount('0')
        ->setTotalTaxAmount('0')
        ->setImageUrl('https://example.com/image_url')
        ->setProductUrl('https://example.com/product_url')
        ->setQuantityUnit('pcs')
        ->addMerchantMarketplaceSellerInfo('Electronic gadgets')
    $items->addItem($item);

    $request->setItems($items);

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>klarna_authorize</transaction_type>
   <transaction_id>43671</transaction_id>
   <notification_url>https://www.example.com/notification</notification_url>
   <return_success_url>http://www.example.com/success</return_success_url>
   <return_failure_url>http://www.example.com/failure</return_failure_url>
   <return_cancel_url>http://www.example.com/cancel</return_cancel_url>
   <usage>40208 concert tickets</usage>
   <remote_ip>245.253.2.12</remote_ip>
   <amount>60</amount>
   <currency>USD</currency>
   <customer_phone>+1987987987987</customer_phone>
   <customer_email>travis@example.com</customer_email>
   <customer_gender>male</customer_gender>
   <order_tax_amount>0</order_tax_amount>
   <billing_address>
     <first_name>Travis</first_name>
     <last_name>Pastrana</last_name>
     <address1>Muster Str. 12</address1>
     <zip_code>10178</zip_code>
     <city>Los Angeles</city>
     <state>CA</state>
     <country>US</country>
   </billing_address>
   <shipping_address>
     <first_name>Travis</first_name>
     <last_name>Pastrana</last_name>
     <address1>Muster Str. 12</address1>
     <zip_code>10001</zip_code>
     <city>Los Angeles</city>
     <state>CA</state>
     <country>US</country>
   </shipping_address>
   <items>
     <item>
       <item_type>physical</item_type>
       <reference>19-402-USA</reference>
       <name>BatteryPowerPack</name>
       <quantity>1</quantity>
       <unit_price>60</unit_price>
       <tax_rate>0</tax_rate>
       <total_amount>60</total_amount>
       <total_discount_amount>0</total_discount_amount>
       <total_tax_amount>0</total_tax_amount>
       <image_url>https://example.com/image_url</image_url>
       <product_url>https://example.com/product_url</product_url>
       <quantity_unit>pcs</quantity_unit>
       <merchant_data>
         <marketplace_seller_info>Electronic gadgets</marketplace_seller_info>
       </merchant_data>
     </item>
   </items>
 </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **klarna_authorize** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |

| Parameter | Required | Format | Description |
|---|---|---|---|
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |
| customer_gender | optional | | Customer gender |
| order_tax_amount | required | | Non-negative, minor units. The total tax amount of the order |
| **items** | required | | List with items |
| item_type | required | string(255) | Order line type. Possible values: Supported item types |
| quantity | required | integer | Non-negative. The item quantity |
| unit_price | required | integer | Minor units. Includes tax, excludes discount(max value: 100000000) |
| total_amount | required | integer | Includes tax and discount. Must match (quantity unit price) - total discount amount divided by quantity (max value: 100000000) |
| reference | optional | string(255) | Article number, SKU or similar |
| name | optional | string(255) | Descriptive item name |
| tax_rate | optional | integer | Non-negative. In percent, two implicit decimals. I.e 2500 = 25.00 percent |
| total_discount_amount | optional | integer | Non-negative minor units. Includes tax |
| total_tax_amount | optional | integer | Must be within 1 of total amount - total_amount * 10000 / (10000 + tax rate). Negative when type is discount |
| image_url | optional | url | URL to an image that can be later embedded in communications between Klarna and the customer. (max 1024 characters) |
| product_url | optional | url | URL to an image that can be later embedded in communications between Klarna and the customer. (max 1024 characters) |
| quantity_unit | optional | string(8) | Unit used to describe the quantity, e.g. kg, pcs... If defined has to be 1-8 characters |
| **product_identifiers** | optional | | List with product identifiers |
| brand | optional | string(255) | The product's brand name as generally recognized by consumers. If no brand is available for a product, do not supply any value |
| category_path | optional | string(255) | The product's category path as used in the merchant's webshop. Include the full and most detailed category and separate the segments with ' > ' |
| global_trade_item_number | optional | string(255) | The product's Global Trade Item Number (GTIN). Common types of GTIN are EAN, ISBN or UPC. Exclude dashes and spaces, where possible |
| manufacturer_part_number | optional | string(255) | The product's Manufacturer Part Number (MPN), which - together with the brand - uniquely identifies a product. Only submit MPNs assigned by a manufacturer and use the most specific MPN possible |
| **merchant_data** | optional | | List with merchant data |
| marketplace_seller_info | optional | string(255) | Information for merchant marketplace |

`required* = conditionally required`

Supported countries:

| Country | Country code |
|---|---|
| Austria | AT |
| Denmark | DK |
| Finland | FI |
| Germany | DE |

| Country | Country code |
|---|---|
| Netherlands | NL |
| Norway | NO |
| Sweden | SE |

Supported item types:

| Item Types |
|---|
| physical |
| discount |
| shipping fee |
| sales tax |
| digital |
| gift card |
| store credit |
| surcharge |

Successful Response

```
stdClass Object
(
    [transaction_type] => klarna_authorize
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:22.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 60
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>klarna_authorize</transaction_type>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
  <timestamp>2020-07-15T12:08:22Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>60</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |

| Parameter | Type | Description |
|---|---|---|
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [transaction_type] => klarna_authorize
    [status] => error
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 340
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:22.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 60
    [currency] => USD
    [sent_to_acquirer] => false
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>klarna_authorize</transaction_type>
  <status>error</status>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>340</code>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:22Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>60</amount>
  <currency>USD</currency>
  <sent_to_acquirer>false</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |

| Parameter | Type | Description |
|---|---|---|
| sent_to_acquirer | string(255) | "true" or "false" |

#### KLARNA CAPTURE

**Klarna capture settles a Klarna transaction.**

Do this when you are shipping goods, for example. A Klarna capture can only be used after aKlarna Authorize on the same transaction.

Therefore, the `reference_id` of the Klarna transaction is mandatory.

> You can also use klarna capture for partial amount of the initial klarna authorize amount but klarna capture amount should be the same as the sum of items total amount. However, you cannot capture a higher amount than initially authorized.

**Transaction workflow:**

1. The merchant sends Klarna transaction to the gateway.
2. The gateway replies to it. One of returned values is the unique id of the transaction.
3. The merchant sends Klarna capture transaction. Its reference id is unique id of Klarna response.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Alternatives\Klarna\Capture');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setReferenceId('43672')
        ->setAmount('60')
        ->setCurrency('USD');

    // Klarna Items
    $items = new \Genesis\API\Request\Financial\Alternatives\Klarna\Items('USD');
    $item  = new \Genesis\API\Request\Financial\Alternatives\Klarna\Item();
    $item
        ->setItemType('physical')
        ->setReference('19-402-USA')
        ->setName('BatteryPowerPack')
        ->setQuantity('1')
        ->setUnitPrice('60')
        ->setTaxRate('0')
        ->setTotalAmount('60')
        ->setTotalDiscountAmount('0')
        ->setTotalTaxAmount('0')
        ->setImageUrl('https://example.com/image_url')
        ->setProductUrl('https://example.com/product_url')
        ->setQuantityUnit('pcs')
        ->addMerchantMarketplaceSellerInfo('Electronic gadgets')
    $items->addItem($item);

    $request->setItems($items);

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>klarna_capture</transaction_type>
   <transaction_id>43671</transaction_id>
   <usage>40208 concert tickets</usage>
   <remote_ip>245.253.2.12</remote_ip>
   <reference_id>43672</reference_id>
   <amount>60</amount>
   <currency>USD</currency>
   <items>
     <item>
       <item_type>physical</item_type>
       <reference>19-402-USA</reference>
       <name>BatteryPowerPack</name>
       <quantity>1</quantity>
       <unit_price>60</unit_price>
       <tax_rate>0</tax_rate>
       <total_amount>60</total_amount>
       <total_discount_amount>0</total_discount_amount>
       <total_tax_amount>0</total_tax_amount>
       <image_url>https://example.com/image_url</image_url>
       <product_url>https://example.com/product_url</product_url>
       <quantity_unit>pcs</quantity_unit>
       <merchant_data>
         <marketplace_seller_info>Electronic gadgets</marketplace_seller_info>
       </merchant_data>
     </item>
   </items>
 </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **klarna_capture** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| reference_id | required | string(32) | Unique id returned by corresponding transaction |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| **items** | required | | List with items |
| item_type | required | string(255) | Order line type. Possible values: Supported item types |
| quantity | required | integer | Non-negative. The item quantity |
| unit_price | required | integer | Minor units. Includes tax, excludes discount(max value: 100000000) |
| total_amount | required | integer | Includes tax and discount. Must match (quantity unit price) - total discount amount divided by quantity (max value: 100000000) |
| reference | optional | string(255) | Article number, SKU or similar |
| name | optional | string(255) | Descriptive item name |
| tax_rate | optional | integer | Non-negative. In percent, two implicit decimals. I.e 2500 = 25.00 percent |
| total_discount_amount | optional | integer | Non-negative minor units. Includes tax |
| total_tax_amount | optional | integer | Must be within 1 of total amount - total_amount * 10000 / (10000 + tax rate). Negative when type is discount |
| image_url | optional | url | URL to an image that can be later embedded in communications between Klarna and the customer. (max 1024 characters) |
| product_url | optional | url | URL to an image that can be later embedded in communications between Klarna and the customer. (max 1024 characters) |
| quantity_unit | optional | string(8) | Unit used to describe the quantity, e.g. kg, pcs... If defined has to be 1-8 characters |
| **product_identifiers** | optional | | List with product identifiers |
| brand | optional | string(255) | The product's brand name as generally recognized by consumers. If no brand is available for a product, do not supply any value |
| category_path | optional | string(255) | The product's category path as used in the merchant's webshop. Include the full and most detailed category and separate the segments with ' > ' |

| Parameter | Required | Format | Description |
|---|---|---|---|
| global_trade_item_number | optional | string(255) | The product's Global Trade Item Number (GTIN). Common types of GTIN are EAN, ISBN or UPC. Exclude dashes and spaces, where possible |
| manufacturer_part_number | optional | string(255) | The product's Manufacturer Part Number (MPN), which - together with the brand - uniquely identifies a product. Only submit MPNs assigned by a manufacturer and use the most specific MPN possible |
| **merchant_data** | optional | | List with merchant data |
| marketplace_seller_info | optional | string(255) | Information for merchant marketplace |

`required* = conditionally required`

Supported countries:

| Country | Country code |
|---|---|
| Austria | AT |
| Denmark | DK |
| Finland | FI |
| Germany | DE |
| Netherlands | NL |
| Norway | NO |
| Sweden | SE |

Supported item types:

| Item Types |
|---|
| physical |
| discount |
| shipping_fee |
| sales_tax |
| digital |
| gift_card |
| store_credit |
| surcharge |

Successful Response

```
stdClass Object
(
    [transaction_type] => klarna_capture
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:22.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 60
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>klarna_capture</transaction_type>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
  <timestamp>2020-07-15T12:08:22Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>60</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```


**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response


```
stdClass Object
(
    [transaction_type] => klarna_capture
    [status] => error
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 340
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:22.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 60
    [currency] => USD
    [sent_to_acquirer] => false
)
```


```
This request is not implemented yet
```

```
This request is not implemented yet
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>klarna_capture</transaction_type>
  <status>error</status>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>340</code>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:22Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>60</amount>
  <currency>USD</currency>
  <sent_to_acquirer>false</sent_to_acquirer>
</payment_transaction>
```


**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| sent_to_acquirer | string(255) | "true" or "false" |

---

# Crypto

Alternative payment methods supporting digital cryptocurrencies.

#### BITPAY SALE

BitPay is a cryptocurrency payments provider supporting blockchain payments with Bitcoin (BTC) and BitcoinCash (BCH).

BitPay Sale is an asynchronous transaction type.

When this payment method is selected at checkout, the customer will be redirected to the BitPay system window including all the data for the payment: Bitcoin/BitcoinCash account, amount to be paid in cryptocurrency and the Fiat equivalent.

If the customer possesses a BitPay wallet or another BitPay compatible crypto wallet, the payment can be done from that window with one click, otherwise a QR CODE containing all the payment data can be scanned and used in any crypto wallet.

Then the customer has 15 minutes to fulfill the generated invoice.

If that timeframe is not met, the invoice will expire and the Merchant will be notified.

If the invoice is fulfilled in the timeframe, it needs to obtain 6 blockchain confirmations (1 hour) before it's safe for the payment to be considered as completed.

At that point, the Merchant will be notified for the approved payment.

After the 6th confirmation, when the transaction is completed, the Merchant can process a refund if it's needed.

> Each transaction has amount limits of minimum 1 USD and maximum of 950 000 USD or its equivalent in EUR

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Crypto\BitPay\Sale');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setReturnUrl('https://staging.gate.e-comprocessing.net/redirect/to_acquirer/a55ab44d242f')
        ->setAmount('3000')
        ->setCurrency('EUR')
        ->setCustomerEmail('travis@example.com')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Berlin')
        ->setBillingCountry('DE');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```curl
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>bitpay_sale</transaction_type>
   <transaction_id>43671</transaction_id>
   <usage>40208 concert tickets</usage>
   <remote_ip>245.253.2.12</remote_ip>
   <return_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/a55ab44d242f</return_url>
   <amount>3000</amount>
   <currency>EUR</currency>
   <customer_email>travis@example.com</customer_email>
   <billing_address>
     <first_name>Travis</first_name>
     <last_name>Pastrana</last_name>
     <address1>Muster Str. 12</address1>
     <zip_code>10178</zip_code>
     <city>Berlin</city>
     <country>DE</country>
   </billing_address>
 </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **bitpay_sale** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | optional | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| return_url | required | url | URL where consumer is sent to after payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| customer_email | required | e-mail address | Must contain valid e-mail of customer |

| Parameter | Required | Format | Description |
|---|---|---|---|
| **billing_address** | required* | | See Required vs Optional API params for details |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

Successful Response

```
stdClass Object
(
    [transaction_type] => bitpay_sale
    [status] => approved
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:22.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 3000
    [currency] => EUR
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>bitpay_sale</transaction_type>
  <status>approved</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
  <timestamp>2020-07-15T12:08:22Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>3000</amount>
  <currency>EUR</currency>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [transaction_type] => bitpay_sale
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 110
    [technical_message] => Something went wrong, please contact support!
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:22.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 3000
    [currency] => EUR
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>bitpay_sale</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>110</code>
  <technical_message>Something went wrong, please contact support!</technical_message>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:22Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>3000</amount>
  <currency>EUR</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |

| Parameter | Type | Description |
|---|---|---|
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

---

# Payouts

**BANK PAY-OUT**

Bank Pay-out is a bank pay-out method. It allows merchants to transfer funds directly to customers' bank accounts.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\OnlineBankingPayments\OnlineBanking\Payout');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setNotificationUrl('https://www.example.com/notification')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setAmount('50000')
        ->setCurrency('INR')
        ->setCustomerPhone('+1987987987987')
        ->setCustomerEmail('travis@example.com')
        ->setBankName('Netbanking')
        ->setBankBranch('HDFC0000001')
        ->setBankAccountNumber('1234123412341234')
        ->setBankAccountName('Anurak Nghuen')
        ->setIdCardNumber('123789456')
        ->setPayerBankPhoneNumber('01234567891')
        ->setBankAccountType('C')
        ->setDocumentType('PASS')


        // Billing Address
        ->setBillingFirstName('Anurak')
        ->setBillingLastName('Nghuen')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('New Delhi')
        ->setBillingState('New Delhi')
        ->setBillingCountry('IN');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>bank_payout</transaction_type>
   <transaction_id>43671</transaction_id>
   <usage>40208 concert tickets</usage>
   <remote_ip>245.253.2.12</remote_ip>
   <notification_url>https://www.example.com/notification</notification_url>
   <return_success_url>http://www.example.com/success</return_success_url>
   <return_failure_url>http://www.example.com/failure</return_failure_url>
   <amount>50000</amount>
   <currency>INR</currency>
   <customer_phone>+1987987987987</customer_phone>
   <customer_email>travis@example.com</customer_email>
   <bank_name>Netbanking</bank_name>
   <bank_branch>HDFC0000001</bank_branch>
   <bank_account_number>1234123412341234</bank_account_number>
   <bank_account_name>Anurak Nghuen</bank_account_name>
   <id_card_number>123789456</id_card_number>
   <payer_bank_phone_number>01234567891</payer_bank_phone_number>
   <bank_account_type>C</bank_account_type>
   <document_type>PASS</document_type>
   <billing_address>
     <first_name>Anurak</first_name>
     <last_name>Nghuen</last_name>
     <address1>Muster Str. 12</address1>
     <zip_code>10178</zip_code>
     <city>New Delhi</city>
     <state>New Delhi</state>
     <country>IN</country>
   </billing_address>
 </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **bank_payout** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| notification_url | required | url | URL at merchant where gateway sends outcome of transaction. |
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| customer_email | required* | e-mail address | Must contain valid e-mail of customer |
| customer_phone | required* | string(32) | Must contain valid phone number of customer |
| bank_name | required* | bank name | Must contain Bank Names |
| bank_branch | required* | bank branch | Name of the Bank branch |
| bank_account_name | required* | bank account name | Bank account name is required, for CNY currency and should be in Simplified Chinese. For other currency, must be in English Language. |
| bank_account_number | required* | bank account number | Bank account number of the customer. |
| bank_province | required* | bank province | Name of the province that the bank is located. |
| id_card_number | required* | string(30) | ID card number |
| payer_bank_phone_number | required* | string(11) | Payer bank phone number |
| bank_account_type | required* | string(1) | The type of account. C: for Checking accounts S: for Savings accounts M: for Maestra accounts(Only Peru) |
| document_type | required* | string(10) | ID card/document type |
| account_id | required* | string(255) | Unique account identifier in Trustly's system. You will receive this after Select Account call and after Trustly Sale on the notification URL. |
| user_id | required* | string(255) | Unique user identifier defined by merchant |

| Parameter | Required | Format | Description |
|---|---|---|---|
| birth_date | required* | dd-mm-yyyy | Required for Visa only when MCC is a Financial Services one (e.g. MCC 6012) |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required | string(255) | Customer first name |
| last_name | required | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

**Supported currencies**

| Currency name | Currency code |
|---|---|
| Argentine peso | ARS |
| Brazilian real | BRL |
| Chilean peso | CLP |
| China yen | CNY |
| Colombian peso | COP |
| Indonesian rupiah | IDR |
| Indian rupee | INR |
| Malaysian ringgit | MYR |
| Mexican peso | MXN |
| Peruvian sol | PEN |
| Thai baht | THB |
| Uruguayan peso | UYU |

**BANK NAMES**

Bank Names may vary based on the specific setup.

**For CNY currency:**

| Bank Name |
|---|
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |

| Bank Name |
| --- |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |
| |

**For MYR currency:**

| Bank Name |
| --- |
| 423 |
| CIMB Clicks Bank |
| Hong Leong Bank |
| May Bank |
| Public Bank |
| RHB Bank |

**For THB currency:**

| Bank Name |
| --- |
| Bangkok Bank |
| Kasikorn Bank |
| Krungsri (Bank of Ayudhya Public Company Limited) |
| Krung Thai Bank |
| Siam Commercial Bank |
| UOBT |

**For IDR currency:**

| Bank Name |
| --- |
| Bank Central Asia |
| Bank Rakyat Indonesia |
| Bank Negara Indonesia |
| BTN Bank |
| CIMB Clicks Indonesia |
| Danamon Bank |
| Mandiri Bank |
| Permata Bank |

**For INR currency:**

| Bank Name |
| --- |
| ABHYUDAYA COOP BANK |
| THE ROYAL BANK OF SCOTLAND |
| ABU DHABI COMMERCIAL BANK |
| THE AKOLA DISTRICT CENTRAL COOPERATIVE BANK |
| AIRTEL PAYMENTS BANK LIMITED |
| AKOLA JANATA COMMERCIAL COOPERATIVE BANK |
| ALLAHABAD BANK |
| THE AHMEDABAD MERC COOP BANK |
| ANDHRA BANK |
| AUSTRALIA & NEW ZEALAND BANK |
| THE ANDHRA PRADESH STATE COOP BANK |
| ANDHRA PRAGATI GRAMEEN BANK |
| THE A.P. MAHESH CO-OP URBAN BANK |
| APNA SAHAKARI BANK LTD |
| ALMORA URBAN CO-OPERATIVE BANK LTD. |
| BASSEIN CATHOLIC CO-OP BANK |
| BANK OF BARODA |
| BARCLAYS BANK |
| BANK OF BAHREIN & KUWAIT |
| THE BHARAT COOPERATIVE BANK |
| BANK OF CEYLON |
| BANDHAN BANK LIMITED |
| DENA BANK |
| BANK OF INDIA |
| BHARATIYA MAHILA BANK LIMITED |
| B N PARIBAS BANK |
| BANK OF AMERICA |
| BANK OF TOKYO-MITSUBISHI |
| CENTRAL BANK OF INDIA |
| CITIZEN CREDIT COOP BANK |
| JP MORGAN CHASE BANK |
| CITI BANK |

| Bank Name |
| --- |
| CITY UNION BANK |
| CAPITAL LOCAL AREA BANK LTD. |
| CANARA BANK |
| CORPORATION BANK |
| THE COSMOS CO-OP. BANK |
| CREDIT SUISSE AG? |
| CREDIT AGRICOLE CORP N INVSMNT BK |
| CHHATRAPATI RAJARSHISHAHU COOP BANK |
| CATHOLIC SYRIAN BANK |
| COMMONWEALTH BK OF AUSTRALIA |
| CHINATRUST COMMERCIAL BANK |
| DEVELOPMENT BANK OF SINGAPORE |
| DEVELOPMENT CREDIT BANK |
| DEOGIRI NAGARI SAHAKARI BANK LTD. AURANGABAD |
| DEUTSCHE BANK |
| DICGC |
| THE DELHI STATE COOPERATIVE BANK LIMITED |
| DHANALAXMI BANK |
| DOMBIVLI NAGARI SAHAKARI BANK LTD |
| DOHA BANK QSC |
| EXPORT IMPORT BANK OF INDIA |
| EQUITAS SMALL FINANCE BANK LIMITED |
| THE FEDERAL BANK |
| FIRSTRAND BANK |
| THE GREATER BOMBAY CO-OP. BANK LTD |
| THE GADCHIROLI DISTRICT CENTRAL COOPERATIVE BANK LIMITED |
| GURGAON GRAMIN BANK LTD. |
| THE GUJARAT STATE CO-OPERATIVE BANK |
| THE HASTI COOP BANK LTD |
| HDFC BANK LTD. |
| HIMACHAL PRADESH STATE COOPERATIVE BANK LTD |
| HONG KONG & SHANGHAI BANK |
| Woori |
| PT BANK MAYBANK INDONESIA TBK |
| IDBI BANK |
| INDUSTRIAL BANK OF KOREA |
| INDUSTRIAL AND COMMERCIAL BANK OF CHINA LIMITED |
| ICICI BANK LTD. |
| IDFC BANK LIMITED |
| INDIAN BANK |
| IDUKKI DISTRICT CO OPERATIVE BANK LTD |
| INDUS-IND BANK |
| INDIAN OVERSEAS BANK |

| Bank Name |
| --- |
| THE JAMMU & KASHMIR BANK |
| JANSEVA SHAHKARI BANK LTD. PUNE |
| JANASEVA SAHAKARI BANK BORIVLI LIMITED |
| JALGAON JANATA SAHAKARI |
| THE JALGAON PEOPELS COOPERATIVE BANK LIMITED |
| JANKALYAN SHAKARI BANK |
| JANATA SAHAKARI BANK LTD (PUNE) |
| THE KANGRA CENTRAL COOPERATIVE BANK |
| KALLAPPANNA AWADE ICH JANATA S |
| THE KANGRA COOPERATIVE BANK LTD |
| KARNATAKA BANK |
| KAPOLE BANK |
| THE KALUPUR COMM COOP BANK |
| THE KALYAN JANATA SAHAKARI BANK |
| KOTAK MAHINDRA BANK |
| KERALA GRAMIN BANK |
| THE KURMANCHAL NAGAR SAHAKARI BANK LIMITED |
| THE KARNATAKA STATE COOP APEX BANK |
| KEB Hana Bank |
| THE KARAD URBAN COOP BANK LTD |
| KARUR VYSYA BANK |
| KARNATAKA GRAMIN VIKAS BANK |
| THE LAKSHMI VILAS BANK |
| BANK OF MAHARASHTRA |
| Maharashtra Gramin Bank |
| MAHANAGAR COOP BANK |
| MUMBAI DISTRICT CENTRAL CO-OP BANK |
| MIZUHO CORPORATE BANK LTD |
| Maharashtra State Cooperative Bank |
| MASHREQ BANK |
| THE MEHSANA URBAN COOPERATIVE BANK |
| THE MUNICIPAL CO OPERATIVE BANK LTD |
| NATIONAL AUSTRALIA BANK LIMITED |
| NATIONAL BANK OF ABU DHABI PJSC |
| NAGPUR NAGRIK (NNSB LTD*) |
| NEW INDIA CO-OPERATIVE BANK |
| NKGSB BANK |
| THE NASIK MERCHANTS CO-OP BANK LTD. |
| NORTH MALBAR GRAMIN BANK |
| NUTAN NAGARIK SAHAKARI BANK |
| THE BANK OF NOVA SCOTIA |
| THE NAINITAL BANK LTD |
| NAGAR URBAN CO OPERATIVE BANK |

| Bank Name |
| --- |
| OMAN INTERNATIONAL BANK |
| ORIENTAL BANK OF COMMERCE |
| PARSIK JANATA SAHAKARI BANK |
| PRAGATHI KRISHNA GRAMIN BANK |
| PUNJAB AND MAHARASHTRA CO-OP BANK |
| PRIME CO OPERATIVE BANK LTD |
| PRATHAMA BANK |
| PUNJAB AND SIND BANK |
| THE PANDHARPUR URBAN CO OP. BANK LTD. PANDHARPUR |
| PUNJAB NATIONAL BANK |
| RABOBANK INTERNATIONAL (CCRB) |
| THE RATNAKAR BANK |
| RESERVE BANK OF INDIA |
| RAJKOT NAGARIK SAHAKARI BANK LTD |
| RAJGURUNAGAR SAHAKARI BANK LIMITED |
| THE RAJASTHAN STATE CO-OP BANK |
| SBERBANK |
| SAHEBRAO DESHMUKH COOPERATIVE BANK LIMITED |
| STATE BANK OF BIKANER AND JAIPUR |
| STATE BANK OF HYDERABAD |
| STATE BANK OF INDIA |
| STATE BANK OF MYSORE |
| SAMARTH SAHAKARI BANK LTD |
| STATE BANK OF TRAVANCORE |
| STANDARD CHARTERED BANK |
| THE SURAT DISTRICT CO-OP BAN |
| SHINHAN BANK |
| SHIKSHAK SAHAKARI BANK LIMITED |
| SOUTH INDIAN BANK |
| SOLAPUR JANATA SAHAKARI BANK LIMITED |
| SUMITOMO MITSUI BANKING CORPORATION |
| SHIVALIK MERCANTILE CO OPERATIVE BANK LTD |
| SOCIETE GENERALE |
| THE SURAT PEOPLE?S CO-OP BANK |
| THE SARASWAT CO-OPERATIVE BANK |
| STATE BANK OF PATIALA |
| STATE BANK OF MAURITIUS |
| SURAT NATIONAL COOPERATIVE BANK LIMITED |
| THE SUTEX COOPERATIVE BANK |
| THE SEVA VIKAS COOPERATIVE BANK LIMITED |
| THE SHAMRAO VITHAL COOP BANK |
| SYNDICATE BANK |
| THANE BHARAT SAHAKARI BANK LTD |

| Bank Name |
| --- |
| THE THANE DISTRICT CENTRAL COOPERATIVE BANK LIMITED |
| TUMKUR GRAIN MERCHANTS CO-OP BANK |
| THE THANE JANATA SAHAKARI BANK |
| TAMILNADU MERC. BANK |
| THE TAMILDADU STATE APEX COOP BANK |
| UNION BANK OF INDIA |
| UBS AG |
| UCO BANK |
| UNITED OVERSEAS BANK LIMITED |
| UNITED BANK OF INDIA |
| AXIS BANK |
| THE VARACHHA CO-OP. BANK LTD. |
| VIJAYA BANK |
| THE VISHWESHWAR SAHAKARI BANK LTD |
| VASAI VIKAS SAHAKARI BANK |
| ING VYSYA BANK |
| THE WEST BENGAL STATE CO-OP BANK |
| WESTPAC BANKING CORPORATION |
| YES BANK |
| THE ZOROASTRIAN COOPERATIVE BANK LIMITED |
| ZILA SAHAKRI BANK LIMITED GHAZIABAD |

**For ARS currency:**

| Bank Name |
| --- |
| A.B.N Amro Bank |
| Banco de Galicia Y Buenos Aires |
| Lloyds Tsb Bank |
| Banco de La Nacion Argentina |
| Banco de La Provincia de Buenos A |
| Bankboston, National Associa |
| Citibank |
| BBVA Banco Frances |
| The Bank Of Tokyo - Mitsubishi |
| Banco de La Provincia de Cordoba |
| Banco Societe Generale |
| Banco de La Ciudad de Buenos Aires |
| Banco Patagonia Sudameris |
| Banco Hipotecario |
| Banco de San Juan |
| Banco Do Brasil |
| Banco Del Tucuman |
| Banco Municipal de Ros |
| Banco Rio de La Plata |
| Banco Regional de Cuyo |

| Bank Name |
| --- |
| Banco Del Chubut |
| Banco de Santa Cruz |
| Banco de La Pampa Sociedad de Economia M |
| Banco de Corrientes |
| Banco Provincia Del Neuquen |
| Banco Empresario de Tucuman Coop. L |
| Banco B. I. Creditanstalt |
| HSBC Bank Argentina |
| J P Morgan Chase Bank Sucursal Buenos Aires |
| Banco Credicoop Coop. L |
| Banco de Valores |
| Banco Roela |
| Banco Mariva |
| Banco Itau Buen Ayre |
| Bank Of America, National Associa |
| Banca Nazionale Del Lavoro |
| Bnp Paribas |
| Banco Provincia de Tierra Del Fuego |
| Banco de La Republica Oriental Del Uruguay |
| Banco Saenz |
| Banco Meridian |
| Banco Macro Bansud |
| Banco Mercurio |
| Ing Bank |
| American Express Bank Ltd |
| Banco Banex |
| Banco Comafi |
| Banco de Inversion Y Comercio Exterior |
| Banco Piano |
| Banco Finansur |
| Banco Julio |
| Banco Privado de Inversiones |
| Nuevo Banco de La Rioja |
| Banco Del Sol |
| Nuevo Banco Del Chaco |
| M. B. A. Banco de Inversiones |
| Banco de Formosa |
| Banco CMF |
| Banco de Santiago Del Estero |
| Nuevo Banco Industrial de Azul |
| Deutsche Bank |
| Nuevo Banco de Santa Fe |
| Banco Cetelem Argentina |

| Bank Name |
| --- |
| Banco de Servicios Financieros |
| Banco Cofidis |
| Banco Bradesco Argentina |
| Banco de Servicios Y Transacciones |
| Rci Ba |
| Bacs Banco de Credito Y Securitizacion |
| Nuevo Banco de Entre Rios |
| Nuevo Banco Suquia |
| Nuevo Banco Bisel |
| Banco Columbia |

**For BRL currency:**

| Bank Name |
| --- |
| BANCO DO BRASIL S.A. |
| BANCO DA AMAZONIA S.A. |
| BANCO DO NORDESTE DO BRASIL S.A. |
| BANESTES S.A. BANCO DO ESTADO DO ESPIRITO SANTO |
| BANCO SANTANDER BRASIL S.A. |
| BANCO DO ESTADO DO PARA S.A. - BANPARA |
| BANCO DO ESTADO DO RIO GRANDE DO SUL S.A. - BANRISUL |
| BANCO DO ESTADO DE SERGIPE S.A. - BANESE |
| BANCO DE BRASILIA S.A. - BRB |
| CAIXA ECONOMICA FEDERAL - CEF |
| BANCO BONSUCESSO S.A. |
| BANCO BRADESCO S.A. |
| BANCO ABC BRASIL S.A. |
| ITAU UNIBANCO S.A. |
| BANCO MERCANTIL DO BRASIL S.A. |
| HSBC BANK BRASIL S.A. - BANCO MULTIPLO |
| BANCO SAFRA S.A. |
| CITIBANK N.A. |
| BANCO DAYCOVAL S.A. |
| BANCO MODAL S.A. |
| BANCO COOPERATIVO SICREDI S.A. |
| BANCO COOPERATIVO DO BRASIL S/A - BANCOOB |

**For CLP currency:**

| Bank Name |
| --- |
| Banco de Chile |
| Banco Internacional |
| Banco del Estado de Chile |
| Scotiabank Chile |
| Banco Crédito e Inversiones |
| Banco Bice |

| Bank Name |
| --- |
| HSBC Bank |
| Banco Santander- Santiago |
| Itau Corpbanca |
| ABN Amro Bank Chile |
| Banco Security |
| Banco Falabella |
| Deutsche Bank |
| Banco Ripley |
| Rabobank Chile |
| Banco Consorcio |
| Banco Penta |
| BBVA Chile |
| Banco del Desarrollo |

**For COP currency:**

| Bank Name |
| --- |
| BANCO DE BOGOTA |
| BANCO POPULAR |
| CORPBANCA |
| BANCOLOMBIA |
| CITIBANK |
| HSBC |
| BANCO SUDAMERIS |
| BBVA |
| HELM BANK S.A |
| BANCO COLPATRIA |
| BANCO DE OCCIDENTE |
| BANCO CAJA SOCIAL BCSC |
| BANCO AGRARIO |
| BANCO DAVIVIENDA |
| BANCO AV VILLAS |
| BANCO PROCREDIT |
| BANCO PICHINCHA |
| BANCOOMEVA |
| BANCO FALABELLA S.A |
| COOPCENTRAL S.A |
| COOPERATIVA FINANCIERA DE ANTIOQUIA |
| COTRAFA COOPERATIVA FINANCIERA |
| CONFIAR |
| FINANCIERA JURISCOOP |

**For MXN currency:**

| Bank Name |
| --- |
| BANAMEX |
| BANCOMEXT |

| Bank Name |
| --- |
| BANOBRAS |
| BBVA BANCOMER |
| SANTANDER |
| BANJERCITO |
| HSBC |
| BAJIO |
| IXE |
| INBURSA |
| INTERACCIONES |
| MIFEL |
| SCOTIABANK |
| BANREGIO |
| INVEX |
| BANSI |
| AFIRME |
| BANORTE |
| THE ROYAL BANK |
| AMERICAN EXPRESS |
| BAMSA |
| TOKYO |
| JP MORGAN |
| BMONEX |
| VE POR MAS |
| ING |
| DEUTSCHE |
| CREDIT SUISSE |
| AZTECA |
| AUTOFIN |
| BARCLAYS |
| COMPARTAMOS |
| BANCO FAMSA |
| BMULTIVA |
| ACTINVER |
| WALMART |
| NAFIN |
| INTERBANCO |
| BANCOPPEL |
| ABC CAPITAL |
| UBS BANK |
| CONSUBANCO |
| VOLKSWAGEN |
| CIBANCO |
| BBASE |

| Bank Name |
| --- |
| BANSEFI |
| HIPOTECARIA FEDERAL |
| MONEXCB |
| GBM |
| MASARI |
| VALUE |
| ESTRUCTURADORES |
| TIBER |
| VECTOR |
| B&B |
| ACCIVAL |
| MERRILL LYNCH |
| FINAMEX |
| VALMEX |
| UNICA |
| MAPFRE |
| PROFUTURO |
| CB ACTINVER |
| OACTIN |
| SKANDIA |
| CBDEUTSCHE |
| ZURICH |
| ZURICHVI |
| SU CASITA |
| CB INTERCAM |
| CI BOLSA |
| BULLTICK CB |
| STERLING |
| FINCOMUN |
| HDI SEGUROS |
| ORDER |
| AKALA |
| CB JPMORGAN |
| REFORMA |
| STP |
| TELECOMM |
| EVERCORE |
| SKANDIA |
| SEGMTY |
| ASEA |
| KUSPIT |
| SOFIEXPRESS |
| UNAGRA |

| Bank Name |
| --- |
| OPCIONES EMPRESARIALES DEL NOROESTE |
| LIBERTAD |
| CLS |
| INDEVAL |

**For PEN currency:**

| Bank Name |
| --- |
| Banco Central de Reserva |
| Banco de Crédito del Perú |
| Interbank |
| Citibank |
| Scotiabank |
| BBVA Continental |
| Banco de la Nación |
| Banco de Comercio |
| Banco Financiero |
| Banco Interamericano de Finanzas (BIF) |
| Crediscotia Financiera |
| Mi Banco |
| Banco GNB Perú S.A. |
| Banco Falabella |
| Santander |
| Caja Metropolitana de Lima |
| Caja Municipal de Ahorro y Crédito Piura SAC |
| Caja Municipal de Ahorro y Crédito Trujillo |
| Caja Municipal de Ahorro y Crédito Arequipa |
| Caja Municipal de Ahorro y Crédito Sullana |
| Caja Municipal de Ahorro y Crédito Cuzco |
| Caja Municipal de Ahorro y Crédito Huancayo |

**For UYU currency:**

| Bank Name |
| --- |
| BROU - Banco de la República Oriental del Uruguay |
| Banco Hipotecario del Uruguay |
| Banco Bandes |
| Banco ITAU |
| Scotiabank |
| Banco Santander |
| Banco Bilbao Vizcaya Argentaria |
| HSBC Bank |
| Banque Heritage |
| Citibank N.A. Sucursal |
| Banco de la Nación Argentina |

Successful Response

```
stdClass Object
(
    [transaction_type] => bank_payout
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:22.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 50000
    [currency] => INR
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>bank_payout</transaction_type>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <timestamp>2020-07-15T12:08:22Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>50000</amount>
  <currency>INR</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [transaction_type] => bank_payout
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 110
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:22.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 50000
    [currency] => INR
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>bank_payout</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>110</code>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:22Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>50000</amount>
  <currency>INR</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

#### BITPAY PAYOUT

BitPay Payout is a crypto currency payout method where merchants are requesting payouts in FIAT currency and the funds are transferred in Bitcoin equivalent to a crypto wallet address.

BitPay Payout is an asynchronous transaction type supported through the Processing API, Virtual Terminal and Web Payment Form.

The payout requests are processed once a day at 11:00 GMT and the settlement usually takes 24 hours.

> For amounts greater than 3000 USD or equivalent in other currency, additional KYC authentication might be required by BitPay

> Each transaction has amount limits of minimum 1 USD and maximum of 950 000 USD or its equivalent in EUR

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Crypto\BitPay\Payout');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setNotificationUrl('https://www.example.com/notification')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setAmount('2000')
        ->setCurrency('USD')
        ->setCryptoAddress('n1jE32b88mtT7UETWvV6WgrV5AUVTPDxW8')
        ->setCryptoWalletProvider('other')
        ->setCustomerEmail('travis@example.com');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```curl
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>bitpay_payout</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <notification_url>https://www.example.com/notification</notification_url>
    <return_success_url>http://www.example.com/success</return_success_url>
    <return_failure_url>http://www.example.com/failure</return_failure_url>
    <amount>2000</amount>
    <currency>USD</currency>
    <crypto_address>n1jE32b88mtT7UETWvV6WgrV5AUVTPDxW8</crypto_address>
    <crypto_wallet_provider>other</crypto_wallet_provider>
    <customer_email>travis@example.com</customer_email>
  </payment_transaction>'
```

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Crypto\BitPay\Payout');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setNotificationUrl('https://www.example.com/notification')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setAmount('500000')
        ->setCurrency('USD')
        ->setCryptoAddress('n1jE32b88mtT7UETWvV6WgrV5AUVTPDxW7')
        ->setCryptoWalletProvider('kraken')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```curl
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>bitpay_payout</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <notification_url>https://www.example.com/notification</notification_url>
    <return_success_url>http://www.example.com/success</return_success_url>
    <return_failure_url>http://www.example.com/failure</return_failure_url>
    <amount>500000</amount>
    <currency>USD</currency>
    <crypto_address>n1jE32b88mtT7UETWvV6WgrV5AUVTPDxW7</crypto_address>
    <crypto_wallet_provider>kraken</crypto_wallet_provider>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>10178</zip_code>
      <city>Los Angeles</city>
      <state>CA</state>
      <country>US</country>
    </billing_address>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **bitpay_payout** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |

| Parameter | Required | Format | Description |
|---|---|---|---|
| notification_url | required | url | URL at merchant where gateway sends outcome of transaction. |
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| customer_email | required* | e-mail address | Must contain valid e-mail of customer |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| crypto_address | required | string(255) | Valid crypto address where the funds will be received |
| crypto_wallet_provider | required | string(255) | If crypto wallet provider is not in the table below, you must send 'other' |
| **billing_address** | required* | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required* | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

**Crypto wallet providers with additional requirements**

| crypto wallet provider | description | website |
|---|---|---|
| BitGo | crypto wallet provider | official website |
| Uphold | crypto wallet provider | official website |
| Circle | crypto exchange | official website |
| Coinbase | crypto wallet provider | official website |
| GDax | crypto exchange | official website |
| Gemini | crypto exchange | official website |
| ITBit | crypto exchange | official website |
| Kraken | crypto exchange | official website |

Address fields are required in case the crypto wallet provider is in the list above and the payout amount is greater than 3000 USD or the equivalent in other currency

Successful Response

```
stdClass Object
(
    [transaction_type] => bitpay_payout
    [status] => approved
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:22.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 500000
    [currency] => USD
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>bitpay_payout</transaction_type>
  <status>approved</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <timestamp>2020-07-15T12:08:22Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>500000</amount>
  <currency>USD</currency>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |

Error Response

```
stdClass Object
(
    [transaction_type] => bitpay_payout
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [response_code] =>
    [code] => 940
    [technical_message] => Bitcoin address is invalid
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:22.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 2000
    [currency] => USD
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```


```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>bitpay_payout</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <payment_transaction>response_code</payment_transaction>
  <code>940</code>
  <technical_message>Bitcoin address is invalid</technical_message>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:22Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>2000</amount>
  <currency>USD</currency>
</payment_transaction>
```


**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| response_code | string(2) | Defines the issuer result of a transaction, the status of a message or some action taken or required. Seeissuer response codes for details |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08- 30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |

#### EZEECARD PAYOUT

eZeeCard Payout is a sync based payout method. It's merchant initiated and can only reference specific transaction types:

- Capture
- Sale
- Sale3d
- InitRecurringSale
- InitRecurringSale3d
- RecurringSale

Those need to have been completed using a card issued by our Issuing API.

> eZeeCard Payout is available through Processing API and VT only!

> eZeeCard Payout has amount limits of minimum 10 EUR and maximum of 800 EUR per transaction or its equivalent in other currencies

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\EzeeCardPayout');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('1000')
        ->setCurrency('EUR')
        ->setReferenceId('43672');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>ezeecard_payout</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>1000</amount>
    <currency>EUR</currency>
    <reference_id>43672</reference_id>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **ezeecard_payout** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| reference_id | required | string(32) | Unique id returned by corresponding transaction |

`required* = conditionally required`

Successful Response

```
stdClass Object
(
    [transaction_type] => ezeecard_payout
    [status] => approved
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [transaction_id] => 43671
    [mode] => live
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:22.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [amount] => 1000
    [currency] => EUR
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>ezeecard_payout</transaction_type>
  <status>approved</status>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <transaction_id>43671</transaction_id>
  <mode>live</mode>
  <timestamp>2020-07-15T12:08:22Z</timestamp>
  <amount>1000</amount>
  <currency>EUR</currency>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |

Error Response

```
stdClass Object
(
    [transaction_type] => ezeecard_payout
    [status] => error
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [transaction_id] => 43671
    [technical_message] => invalid deposit transaction
    [message] =>
    [mode] => live
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:22.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [amount] => 1000
    [currency] => EUR
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>ezeecard_payout</transaction_type>
  <status>error</status>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <transaction_id>43671</transaction_id>
  <technical_message>invalid deposit transaction</technical_message>
  <payment_transaction>message</payment_transaction>
  <mode>live</mode>
  <timestamp>2020-07-15T12:08:22Z</timestamp>
  <amount>1000</amount>
  <currency>EUR</currency>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |

#### PAYOUT

Payouts are credits without a reference transaction and as such are highly regulated and need specific gateway terminal configuration, so be sure to contact the IT Support team at tech-support@e-comprocessing.com in case you want Payouts to be enabled. A valid bank account number needs to be provided.

Using a payout, the amount is billed to the customer's credit card. It can be reversed via a void transaction on the same day of the transaction.

Both Visa and Mastercard/Maestro Payouts are authorized real-time.

Note that for exceptional cases with some countries Visa OCTS will not be authorized through the schemes but batched for offline settlement on the same day. This means that the authorization code and issuer response code will not be available only for them.

This transaction type supports Tokenization.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Payout');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('4200000000000000')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.PayoutRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    PayoutRequest request = new PayoutRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("4200000000000000");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.payout(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "4200000000000000",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>payout</transaction_type>
   <transaction_id>43671</transaction_id>
   <usage>40208 concert tickets</usage>
   <remote_ip>245.253.2.12</remote_ip>
   <amount>100</amount>
   <currency>USD</currency>
   <card_holder>Travis Pastrana</card_holder>
   <card_number>4200000000000000</card_number>
   <expiration_month>12</expiration_month>
   <expiration_year>2021</expiration_year>
   <cvv>834</cvv>
   <customer_email>travis@example.com</customer_email>
   <customer_phone>+1987987987987</customer_phone>
   <billing_address>
     <first_name>Travis</first_name>
     <last_name>Pastrana</last_name>
     <address1>Muster Str. 12</address1>
     <zip_code>10178</zip_code>
     <city>Los Angeles</city>
     <state>CA</state>
     <country>US</country>
   </billing_address>
 </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **payout** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| gaming | optional | "true" | Signifies whether a gaming transaction is performed. Gaming transactions usually use MCC 7995. Contacttech-support@e-comprocessing.com for more details |
| moto | optional | "true" | Signifies whether a MOTO (mail order telephone order) transaction is performed. Contacttech-support@e-comprocessing.com for more details |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| card_holder | required | string(255) | Full name of customer as printed on credit card (first name and last name at least) |
| card_number | required | 13 to 16 digits | Complete cc number of customer |
| cvv | required* | 3 to 4 digits | cvv of cc, requirement is based on terminal configuration |
| expiration_month | required | MM | Expiration month as printed on credit card |
| expiration_year | required | YYYY | Expiration year as printed on credit card |
| token | optional | string(36) | See Tokenization for more details. If present, the cardholder parameters can be omitted. Cannot be set together with `remember_card` |
| remember_card | optional | "true" | See Tokenize. Tokenizes cardholder parameters. Cannot be set together with `token` |
| consumer_id | optional | string(10) | See Consumers and Tokenization. Combine with `remember_card` to tokenize or with `token` to use token |
| source_of_funds | optional | string | Specify the source of funds with one of `credit`, `debit`, `prepaid`, `cash`, `other_debit_account`, `other_credit_account`. |
| **credential_on_file** | required* | | See Credential On File (COF) for more details |
| initial_customer_initiated | required* | string(18) | Initial transaction initiated by customer. Required for external tokenization, and optional for gateway-based tokenization |
| subsequent_customer_initiated | required* | string(18) | Subsequent transaction initiated by customer. Required for external tokenization, and optional for gateway-based tokenization |
| merchant_unscheduled | optional | string(20) | Transaction is initiated by the merchant |
| customer_email | required* | e-mail address | Must contain valid e-mail of customer |
| customer_phone | required* | string(32) | Must contain valid phone number of customer |

| Parameter | Required | Format | Description |
|---|---|---|---|
| birth_date | required* | dd-mm-yyyy | Required when MCC is a Financial Services one (e.g. MCC 6012) and either card brand is Visa or Mastercard/Maestro with UK-based merchant, UK-based bin (domestic), and DEBIT card type |
| document_id | required* | sring(20) | Document ID of the consumer. See Document ID Parameter for more details. |
| fx_rate_id | optional | integer | See Get rates for FX Service. Offers the option to use a specific FX rate to convert the transaction processing amount. Used FX rate should have the same source currency as the processing currency. Contact tech-support@e-comprocessing.com for more details |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required* | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |
| **dynamic_descriptor_params** | optional | | |
| merchant_name | optional | string(25) | Allows to dynamically override the charge descriptor |
| merchant_city | optional | string(13) | Allows to dynamically override the merchant phone number |
| sub_merchant_id | optional | string(15) | Allows to dynamically override the sub-merchant ID. |

`required* = conditionally required`

Successful Response

```
stdClass Object
(
    [transaction_type] => payout
    [status] => approved
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [consumer_id] => 123456
    [token] => ee946db8-d7db-4bb7-b608-b65b153e127d
    [avs_response_code] => 5I
    [avs_response_text] => Response provided by issuer processor; Address information not verified
    [authorization_code] => 345678
    [retrieval_reference_number] => 016813015184
    [response_code] => 00
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:22.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[payout]>
    <status content=[approved]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <consumer_id content=[123456]>
    <token content=[ee946db8-d7db-4bb7-b608-b65b153e127d]>
    <avs_response_code content=[5I]>
    <avs_response_text content=[Response provided by issuer processor; Address information not verified]>
    <authorization_code content=[345678]>
    <retrieval_reference_number content=[016813015184]>
    <response_code content=[00]>
    <timestamp content=[2020-07-15T12:08:22Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[true]>
]>
```

```
{
    transaction_type: "payout",
    status: "approved",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    consumer_id: "123456",
    token: "ee946db8-d7db-4bb7-b608-b65b153e127d",
    avs_response_code: "5I",
    avs_response_text: "Response provided by issuer processor; Address information not verified",
    authorization_code: "345678",
    retrieval_reference_number: "016813015184",
    response_code: "00",
    timestamp: "2020-07-15T12:08:22Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "true",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>payout</transaction_type>
  <status>approved</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <consumer_id>123456</consumer_id>
  <token>ee946db8-d7db-4bb7-b608-b65b153e127d</token>
  <avs_response_code>5I</avs_response_code>
  <avs_response_text>Response provided by issuer processor; Address information not verified</avs_response_text>
  <authorization_code>345678</authorization_code>
  <retrieval_reference_number>016813015184</retrieval_reference_number>
  <response_code>00</response_code>
  <timestamp>2020-07-15T12:08:22Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| consumer_id | string(10) | Consumer unique reference. See Consumers |
| token | string(36) | Plain-text token value. See Tokenization |
| avs_response_code | string(255) | Generated by the card network on trying to match the billing address when performing the address verification. Optional, returned if config is enabled and acquirer supports it. Check AVS Status Codes for details. |
| avs_response_text | string(255) | Gives the human response text for the AVS response code above. Optional, returned if config is enabled and acquirer supports it. Check AVS Status Codes for details. |
| authorization_code | string(6) | Generated by the card network when an authorisation has occurred, used to identify that auth. Consists of 6 alphanumeric chars |
| retrieval_reference_number | string(255) | A reference number used for tracking all messages related to a given cardholder transaction returned by some acquirers. |
| response_code | string(2) | Defines the issuer result of a transaction, the status of a message or some action taken or required. See issuer response codes for details |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |

| Parameter | Type | Description |
|---|---|---|
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08- 30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [transaction_type] => payout
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [response_code] => 57
    [code] => 340
    [technical_message] => billing_address[zip_code] is invalid!
    [message] => billing_address[zip_code] is invalid!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:22.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => false
)
```

```
<payment_response content=[
    <transaction_type content=[payout]>
    <status content=[error]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <response_code content=[57]>
    <code content=[340]>
    <technical_message content=[billing_address[zip_code] is invalid!]>
    <message content=[billing_address[zip_code] is invalid!]>
    <timestamp content=[2020-07-15T12:08:22Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[false]>
]>
```

```
{
    transaction_type: "payout",
    status: "error",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    response_code: "57",
    code: "340",
    technical_message: "billing_address[zip_code] is invalid!",
    message: "billing_address[zip_code] is invalid!",
    timestamp: "2020-07-15T12:08:22Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "false",
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>payout</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <response_code>57</response_code>
  <code>340</code>
  <technical_message>billing_address[zip_code] is invalid!</technical_message>
  <message>billing_address[zip_code] is invalid!</message>
  <timestamp>2020-07-15T12:08:22Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>false</sent_to_acquirer>
</payment_transaction>
```


**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| response_code | string(2) | Defines the issuer result of a transaction, the status of a message or some action taken or required. Seeissuer response codes for details |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08- 30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

**MONEY TRANSFER PAYOUT**

Money transfer payout is a standard payout with additional parameters. The section and parameters below are optional and to be considered only when present.

Money transfers: account_to_account, person_to_person, wallet_transfer.

The transaction is not a result of a business operation but rather a pure funds movement from one account (card or non-card) to another (card).

Bear in mind that the sender of the funds in this case is not a merchant, but a consumer.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Payout');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('4200000000000000')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')
        ->setMoneyTransfer('{"type"=>"account_to_account", "sender_account_number"=>"DE91100000000123456789", "sender_address"=>{"first_name"=>"John", "last_name"=>"Smith", "country"=>"JoDEhn", "city"=>"Berlin

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.PayoutRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    PayoutRequest request = new PayoutRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("4200000000000000");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");
    request.setMoneyTransfer("{"type"=>"account_to_account", "sender_account_number"=>"DE91100000000123456789", "sender_address"=>{"first_name"=>"John", "last_name"=>"Smith", "country"=>"JoDEhn", "city"=>"Berl

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.payout(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "4200000000000000",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    },
    "money_transfer": {
      "type": "account_to_account",
      "sender_account_number": "DE91100000000123456789",
      "sender_address": {
        "first_name": "John",
        "last_name": "Smith",
        "country": "JoDEhn",
        "city": "Berlin",
        "zip_code": "10115",
        "address1": "1 Kaiserdamm Blvd, Berlin"
      }
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>payout</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>100</amount>
    <currency>USD</currency>
    <card_holder>Travis Pastrana</card_holder>
    <card_number>4200000000000000</card_number>
    <expiration_month>12</expiration_month>
    <expiration_year>2021</expiration_year>
    <cvv>834</cvv>
    <customer_email>travis@example.com</customer_email>
    <customer_phone>+1987987987987</customer_phone>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>10178</zip_code>
      <city>Los Angeles</city>
      <state>CA</state>
      <country>US</country>
    </billing_address>
    <money_transfer>
      <type>account_to_account</type>
      <sender_account_number>DE91100000000123456789</sender_account_number>
      <sender_address>
        <first_name>John</first_name>
        <last_name>Smith</last_name>
        <country>JoDEhn</country>
        <city>Berlin</city>
        <zip_code>10115</zip_code>
        <address1>1 Kaiserdamm Blvd, Berlin</address1>
      </sender_address>
    </money_transfer>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| **money_transfer** | optional | | Money transfer Parameters |
| type | required | string | The type of money transfer. It can be account_to_account, person_to_person, wallet_transfer |
| sender_account_number | required | string(33) | Sender account number |
| **sender_address** | required | | |
| first_name | required | string(255) | Sender first name |
| last_name | required | string(255) | Sender last name |
| country | required | string(2) | Sender Country code in ISO 3166 |
| city | required | string(255) | Sender City |
| zip_code | required | string | Sender ZIP code |
| address1 | required | string(255) | Sender Primary address |
| state | required* | string(2) | Sender State code in ISO 3166-2, required for USA and Canada |

`required* = conditionally required`

The Sender name is split in two fields 'first name' and 'last name'. Maximum length of these fields must be 24 characters. Ex. 'John Doe' with space between them must not exceed 25 characters.

Money transfer is supported only by Visa.

Different money transfer types are allowed for specific merchant category codes (MCCs).

| Money transfer type | Merchant category codes |
|---|---|
| account_to_account | 4829, 6012 |
| person_to_person | 4829, 6012 |
| wallet_transfer | 4829 |

Successful Response

```
stdClass Object
(
    [transaction_type] => payout
    [status] => approved
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [response_code] =>
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:22.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
)
```

```
<payment_response content=[
    <transaction_type content=[payout]>
    <status content=[approved]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <response_code content=[]>
    <timestamp content=[2020-07-15T12:08:22Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
]>
```

```
{
    transaction_type: "payout",
    status: "approved",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    response_code: "",
    timestamp: "2020-07-15T12:08:22Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>payout</transaction_type>
  <status>approved</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <payment_transaction>response_code</payment_transaction>
  <timestamp>2020-07-15T12:08:22Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
</payment_transaction>
```


Error Response


```
stdClass Object
(
    [transaction_type] => payout
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 340
    [technical_message] => money_transfer_type is not supported. Please select one of account_to_account, person_to_person, wallet_transfer
    [message] => Please check input data for errors!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:22.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
)
```

```
<payment_response content=[
    <transaction_type content=[payout]>
    <status content=[error]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <code content=[340]>
    <technical_message content=[money_transfer_type is not supported. Please select one of account_to_account, person_to_person, wallet_transfer]>
    <message content=[Please check input data for errors!]>
    <timestamp content=[2020-07-15T12:08:22Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
]>
```

```
{
    transaction_type: "payout",
    status: "error",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    code: "340",
    technical_message: "money_transfer_type is not supported. Please select one of account_to_account, person_to_person, wallet_transfer",
    message: "Please check input data for errors!",
    timestamp: "2020-07-15T12:08:22Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>payout</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>340</code>
  <technical_message>money_transfer_type is not supported. Please select one of account_to_account, person_to_person, wallet_transfer</technical_message>
  <message>Please check input data for errors!</message>
  <timestamp>2020-07-15T12:08:22Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
</payment_transaction>
```

**NON-MONEY TRANSFER PAYOUT**

Non-money transfer payout is a standard payout with additional parameters. The section and parameters below are optional and to be considered only when present.

Non-money transfer types are: b2b_supplier, loyalty, funds_disbursment, merchant_settlement, prepaid_card_load.

The transaction is a result of a business operation.

Bear in mind that the sender of the funds in this case is a merchant.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Payout');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('4200000000000000')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')
        ->setMoneyTransfer('{"type"=>"loyalty"}');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.PayoutRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    PayoutRequest request = new PayoutRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("4200000000000000");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");
    request.setMoneyTransfer("{"type"=>"loyalty"}");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.payout(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "4200000000000000",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    },
    "money_transfer": {
      "type": "loyalty"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>payout</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>100</amount>
    <currency>USD</currency>
    <card_holder>Travis Pastrana</card_holder>
    <card_number>4200000000000000</card_number>
    <expiration_month>12</expiration_month>
    <expiration_year>2021</expiration_year>
    <cvv>834</cvv>
    <customer_email>travis@example.com</customer_email>
    <customer_phone>+1987987987987</customer_phone>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>10178</zip_code>
      <city>Los Angeles</city>
      <state>CA</state>
      <country>US</country>
    </billing_address>
    <money_transfer>
      <type>loyalty</type>
    </money_transfer>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| **money_transfer** | optional | | Non-money transfer Parameters |
| type | required | string | The type of non-money transfer. It can be b2b_supplier, loyalty, funds_disbursment, merchant_settlement, prepaid_card_load |

`required* = conditionally required`

Non-money transfer is supported only by Visa.

The listed non-money transfer types are allowed for all merchant category codes (MCCs).

| Non-money transfer type |
|---|

| Non-money transfer type |
| --- |
| b2b_supplier |
| loyalty |
| funds_disbursment |
| merchant_settlement |
| prepaid_card_load |

Successful Response

```
stdClass Object
(
    [transaction_type] => payout
    [status] => approved
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [response_code] =>
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:22.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
)
```

```
<payment_response content=[
    <transaction_type content=[payout]>
    <status content=[approved]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <response_code content=[]>
    <timestamp content=[2020-07-15T12:08:22Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
]>
```

```
{
    transaction_type: "payout",
    status: "approved",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    response_code: "",
    timestamp: "2020-07-15T12:08:22Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
}
```

```
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>payout</transaction_type>
    <status>approved</status>
    <mode>live</mode>
    <transaction_id>43671</transaction_id>
    <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
    <payment_transaction>response_code</payment_transaction>
    <timestamp>2020-07-15T12:08:22Z</timestamp>
    <descriptor>Descriptor one</descriptor>
    <amount>100</amount>
    <currency>USD</currency>
  </payment_transaction>
```

Error Response

```
stdClass Object
(
    [transaction_type] => payout
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 340
    [technical_message] => money_transfer_type is not supported. Please select one of b2b_supplier, loyalty, funds_disbursment, merchant_settlement, prepaid_card_load
    [message] => Please check input data for errors!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:23.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
)
```

```
<payment_response content=[
    <transaction_type content=[payout]>
    <status content=[error]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <code content=[340]>
    <technical_message content=[money_transfer_type is not supported. Please select one of b2b_supplier, loyalty, funds_disbursment, merchant_settlement, prepaid_card_load]>
    <message content=[Please check input data for errors!]>
    <timestamp content=[2020-07-15T12:08:23Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
]>
```

```
{
    transaction_type: "payout",
    status: "error",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    code: "340",
    technical_message: "money_transfer_type is not supported. Please select one of b2b_supplier, loyalty, funds_disbursment, merchant_settlement, prepaid_card_load",
    message: "Please check input data for errors!",
    timestamp: "2020-07-15T12:08:23Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>payout</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>340</code>
  <technical_message>money_transfer_type is not supported. Please select one of b2b_supplier, loyalty, funds_disbursment, merchant_settlement, prepaid_card_load</technical_message>
  <message>Please check input data for errors!</message>
  <timestamp>2020-07-15T12:08:23Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
</payment_transaction>
```

**SCT PAYOUT**

SCT payouts are Sepa-based payouts to consumers done without a reference transaction and as such are regulated and need specific gateway terminal configuration, so be sure to contact the IT Support team at tech-support@e-comprocessing.com) in case you want SCT payouts to be enabled.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\SDD\Payout');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('EUR')
        ->setIban('DE09100100101234567891')
        ->setBic('PBNKDEFFXXX')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingCountry('DE');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.sdd.SDDPayoutRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    SDDPayoutRequest request = new SDDPayoutRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("EUR");
    request.setIban("DE09100100101234567891");
    request.setBic("PBNKDEFFXXX");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingCountry("DE");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.sct_payout(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "EUR",
    "iban": "DE09100100101234567891",
    "bic": "PBNKDEFFXXX",
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "country": "DE"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>sct_payout</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>100</amount>
    <currency>EUR</currency>
    <iban>DE09100100101234567891</iban>
    <bic>PBNKDEFFXXX</bic>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <country>DE</country>
    </billing_address>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| iban | required | string(34) | Customer's IBAN number |
| bic | required | string(11) | SWIFT/BIC code of the customer's bank |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required | string(255) | Customer first name |
| last_name | required | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |

| Parameter | Required | Format | Description |
|---|---|---|---|
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **sct_payout** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| iban | required | string(34) | Customer's IBAN number |
| bic | required | string(11) | SWIFT/BIC code of the customer's bank |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required | string(255) | Customer first name |
| last_name | required | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

**Supported countries:**

Allowed countries include supported countries for SDD Init Recurring Sale and the list below:

| Country Name | Country Code |
|---|---|
| Bulgaria | BG |
| Croatia | HR |

| Country Name | Country Code |
|---|---|
| Czech Republics | CZ |
| Denmark | DK |
| United Kingdom | UK |
| Hungary | HU |
| Iceland | IS |
| Liechtenstein | LI |
| Norway | NO |
| Poland | PL |
| Romania | RO |
| Sweden | SE |
| Switzerland | CH |

Successful Response

```
stdClass Object
(
    [transaction_type] => sct_payout
    [status] => approved
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:23.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => EUR
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[sct_payout]>
    <status content=[approved]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <timestamp content=[2020-07-15T12:08:23Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[EUR]>
    <sent_to_acquirer content=[true]>
]>
```

```
{
    transaction_type: "sct_payout",
    status: "approved",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    timestamp: "2020-07-15T12:08:23Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "EUR",
    sent_to_acquirer: "true",
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>sct_payout</transaction_type>
  <status>approved</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <timestamp>2020-07-15T12:08:23Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>EUR</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```


**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response


```
stdClass Object
(
    [transaction_type] => sct_payout
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 340
    [technical_message] => expiration_year is invalid
    [message] => expiration_year is invalid
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:23.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => EUR
    [sent_to_acquirer] => false
)
```



```
<payment_response content=[
    <transaction_type content=[sct_payout]>
    <status content=[error]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <code content=[340]>
    <technical_message content=[expiration_year is invalid]>
    <message content=[expiration_year is invalid]>
    <timestamp content=[2020-07-15T12:08:23Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[EUR]>
    <sent_to_acquirer content=[false]>
]>
```

```
{
    transaction_type: "sct_payout",
    status: "error",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    code: "340",
    technical_message: "expiration_year is invalid",
    message: "expiration_year is invalid",
    timestamp: "2020-07-15T12:08:23Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "EUR",
    sent_to_acquirer: "false",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>sct_payout</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>340</code>
  <technical_message>expiration_year is invalid</technical_message>
  <message>expiration_year is invalid</message>
  <timestamp>2020-07-15T12:08:23Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>EUR</currency>
  <sent_to_acquirer>false</sent_to_acquirer>
</payment_transaction>
```


**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

**TRUSTLY WITHDRAWAL**

Trustly Withdrawal is an oBeP-style alternative payment method that allows you to transfer money directly to your customers ebank account.

After initiating a transaction Trustly will redirect the customer to their page.

There they have to select their bank and log in with their regular access codes.

Then the customer has to choose the account where they wish to receive the payment (for example, their savings account or current account).

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Alternatives\Trustly\Withdrawal');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')
        ->setBirthDate('21-10-1980')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('12114')
        ->setBillingCity('Berlin')
        ->setBillingCountry('DE');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.apm.TrustlyWithdrawalRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;
import java.net.MalformedURLException;
import java.net.URL;

public class GenesisExample {
  public static void main() throws MalformedURLException {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    TrustlyWithdrawalRequest request = new TrustlyWithdrawalRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setReturnSuccessUrl(new URL("http://www.example.com/success"));
    request.setReturnFailureUrl(new URL("http://www.example.com/failure"));
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");
    request.setBirthDate("21-10-1980");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("12114");
    request.setBillingCity("Berlin");
    request.setBillingCountry("DE");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.trustly_withdrawal(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "return_success_url": "http://www.example.com/success",
    "return_failure_url": "http://www.example.com/failure",
    "amount": "100",
    "currency": "USD",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "birth_date": "21-10-1980",
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "12114",
      "city": "Berlin",
      "country": "DE"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>trustly_withdrawal</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <return_success_url>http://www.example.com/success</return_success_url>
    <return_failure_url>http://www.example.com/failure</return_failure_url>
    <amount>100</amount>
    <currency>USD</currency>
    <customer_email>travis@example.com</customer_email>
    <customer_phone>+1987987987987</customer_phone>
    <birth_date>21-10-1980</birth_date>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>12114</zip_code>
      <city>Berlin</city>
      <country>DE</country>
    </billing_address>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| customer_email | required | e-mail address | Must contain valid e-mail of customer |
| customer_phone | required* | string(32) | Must contain valid phone number of customer |
| birth_date | required | dd-mm-yyyy | Required for Visa only when MCC is a Financial Services one (e.g. MCC 6012) |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |

| Parameter | Required | Format | Description |
|---|---|---|---|
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

Supported countries:

| Country name | Country code |
|---|---|
| Austria | AT |
| Bulgaria | BG |
| Belgium | BE |
| Croatia | HR |
| Czech Republic | CZ |
| Cyprus | CY |
| Denmark | DK |
| Estonia | EE |
| Finland | FI |
| France | FR |
| Germany | DE |
| Greece | GR |
| Hungary | HU |
| Italy | IT |
| Ireland | IE |
| Lithuania | LT |
| Latvia | LV |
| Luxembourg | LU |
| Malta | MT |
| Norway | NO |
| Netherlands | NL |
| Portugal | PT |
| Poland | PL |
| Romania | RO |
| Slovakia | SK |
| Slovenia | SI |
| Sweden | SE |
| Spain | ES |
| United Kingdom | GB |

Successful Response

```
stdClass Object
(
    [transaction_type] => trustly_withdrawal
    [status] => pending_async
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [redirect_url] => https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:23.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[trustly_withdrawal]>
    <status content=[pending_async]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <redirect_url content=[https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <timestamp content=[2020-07-15T12:08:23Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[true]>
]>
```

```
{
    transaction_type: "trustly_withdrawal",
    status: "pending_async",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    redirect_url: "https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    timestamp: "2020-07-15T12:08:23Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "true",
}
```

```
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>trustly_withdrawal</transaction_type>
    <status>pending_async</status>
    <mode>live</mode>
    <transaction_id>43671</transaction_id>
    <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
    <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/649e1ff35c61</redirect_url>
    <technical_message>Transaction successful!</technical_message>
    <message>Transaction successful!</message>
    <timestamp>2020-07-15T12:08:23Z</timestamp>
    <descriptor>Descriptor one</descriptor>
    <amount>100</amount>
    <currency>USD</currency>
    <sent_to_acquirer>true</sent_to_acquirer>
  </payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |

| Parameter | Type | Description |
|---|---|---|
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [transaction_type] => trustly_withdrawal
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 110
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:23.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => false
)
```

```
<payment_response content=[
    <transaction_type content=[trustly_withdrawal]>
    <status content=[error]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <code content=[110]>
    <message content=[Something went wrong, please contact support!]>
    <timestamp content=[2020-07-15T12:08:23Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[false]>
]>
```

```
{
    transaction_type: "trustly_withdrawal",
    status: "error",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    code: "110",
    message: "Something went wrong, please contact support!",
    timestamp: "2020-07-15T12:08:23Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "false",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>trustly_withdrawal</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>110</code>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:23Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>false</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |

| Parameter | Type | Description |
|---|---|---|
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

#### AFRICAN MOBILE PAYOUT

African Mobile Payout, or otherwise known as Disbursement, is an APM used to process Mobile network operator payments. It is an async payment method and will be approved once the payment is processed with the Mobile network operator

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>african_mobile_payout</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <return_success_url>http://www.example.com/success</return_success_url>
    <return_failure_url>http://www.example.com/failure</return_failure_url>
    <amount>100</amount>
    <currency>KES</currency>
    <customer_email>barney.rubble@example.com</customer_email>
    <customer_phone>254701123456</customer_phone>
    <operator>SAFARICOM</operator>
    <target>000010</target>
    <billing_address>
      <first_name>Barney</first_name>
      <last_name>Rubble</last_name>
      <address1>14, Nerazdelni str</address1>
      <zip_code>1407</zip_code>
      <city>Nairobi</city>
      <country>KE</country>
    </billing_address>
    <risk_params>
      <user_id>123456</user_id>
    </risk_params>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **african_mobile_payout** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | required* | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |

| Parameter | Required | Format | Description |
|---|---|---|---|
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| operator | required | string(20) | Name of the Mobile network operator (MNO) which should process the transaction |
| target | required | string(20) | Number of the Paybill for which the transaction is intended |
| customer_phone | required | string(32) | Must contain valid phone number of customer |
| customer_email | required* | e-mail address | Must contain valid e-mail of customer |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

**Supported countries, currencies, operators and payment method:**

| Country code | Currency code | Operator | Payment method |
|---|---|---|---|
| GH | GHS | AIRTEL | Airtel Money |
| GH | GHS | MTN | MTN Mobile Money |
| GH | GHS | TIGO | Tigo Cash |
| GH | GHS | VODACOM | M-PESA |
| KE | KES | AIRTEL | Airtel Money |
| KE | KES | SAFARICOM | M-PESA |
| MZ | MZN | MOVITEL | e-Mola |
| MZ | MZN | VODACOM | M-PESA |
| RW | RWF | MTN | MTN Mobile Money |
| RW | RWF | TIGO | Tigo Cash |
| TZ | TZS | AIRTEL | Airtel Money |
| TZ | TZS | TIGO | Tigo Cash |

| Country code | Currency code | Operator | Payment method |
|---|---|---|---|
| TZ | TZS | VODACOM | M-PESA |
| UG | UGX | AIRTEL | Airtel Money |
| UG | UGX | MTN | MTN Mobile Money |

Successful Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>african_mobile_payout</transaction_type>
  <status>pending_async</status>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <timestamp>2020-07-15T12:08:23Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>KES</currency>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |

Error Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>african_mobile_payout</transaction_type>
  <status>error</status>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <message>Something went wrong, please contact support!</message>
  <technical_message>operator is not supported!</technical_message>
  <timestamp>2020-07-15T12:08:23Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>KES</currency>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| message | string(255) | Human readable error message which can be displayed to users. |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |

#### TRANSFERTO PAYOUT

TransferTo Payout is an APM which provides 3 different payment services: BankAccount, MobileWallet and CashPickup. Merchant sends money to a consumer. Money are delivered through a Payer institution which supports one of the 3 services and has specific requirements on the transaction's amount and required fields. The process is async and once the TransferTo processes the transaction, a notification is received and the status is updated.

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>transfer_to_payout</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>Funding</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <return_success_url>http://www.example.com/success</return_success_url>
    <return_failure_url>http://www.example.com/failure</return_failure_url>
    <amount>100</amount>
    <currency>USD</currency>
    <customer_email>barney.rubble@example.com</customer_email>
    <payer_id>7</payer_id>
    <bank_account_number>0842024000</bank_account_number>
    <billing_address>
      <last_name>Rubble</last_name>
    </billing_address>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **transfer_to_payout** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |

| Parameter | Required | Format | Description |
|---|---|---|---|
| customer_email | required* | e-mail address | Must contain valid e-mail of customer |
| payer_id | required | string | ID of the Payer used to deliver the money through one of the 3 services |
| bank_account_number | required* | string | Bank identification number of the customer. *Requirement based on the Payer |
| ifs_code | required* | string | Bank code of the bank in which the consumer resides. *Requirement based on the Payer |
| msisdn | required* | string | Number uniquely identifying a subscription in a Global System. *Requirement based on the Payer |
| branch_number | required* | string | Branch number. *Requirement based on the Payer |
| account_type | required* | string | Account type. *Requirement based on the Payer |
| registered_name | required* | string | Registered name of the business. *Requirement based on the Payer |
| registration_number | required* | string | Registration number. *Requirement based on the Payer |
| iban | required* | string | Bank account number in IBAN format. *Requirement based on the Payer |
| id_type | required* | string | Identification type. Allowed values: PASSPORT, NATIONAL_ID, DRIVING_LICENSE, SOCIAL_SECURITY, TAX_ID, SENIOR_CITIZEN_ID, BIRTH_CERTIFICATE, VILLAGE_ELDER_ID, RESIDENT_CARD, ALIEN_REGISTRATION, PAN_CARD, VOTERS_ID, HEALTH_CARD, EMPLOYER_ID, OTHER. *Requirement based on the Payer |
| id_number | required* | string | Identification number. *Requirement based on the Payer |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

**Supported currencies:**

| Currency |
|---|
| EUR |
| GBP |
| HKD |
| USD |

Successful Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>transfer_to_payout</transaction_type>
  <status>pending_async</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <timestamp>2020-07-15T12:08:23Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |

Error Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>transfer_to_payout</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>910</code>
  <technical_message>Payer is currently unavailable</technical_message>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:23Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |

# Mobile Payments

**APPLE PAY**

Apple Pay is a mobile payment solution available on iOS devices with Touch ID / Face ID support. Apple Pay allows shoppers to purchase with credit and debit cards linked to their devices.

Payment Object structure

```
{
  "token": {
    "paymentData": { ... },
    "paymentMethod": { ... },
    "transactionIdentifier": "32b...4f3"
  },
  "billingContact": { ... },
  "shippingContact": { ... }
}
```

To use the Apple Pay, your application should be set up with public, private keys and payment processing certificate. Please contactech-support@e-comprocessing.com to enable Apple Pay payments.

Once a payment authorized by the customer in the merchant's application, the Apple Pay APIs will return a Payment Object containing the payment data (Encrypted Payment Token) with customer information to the merchant's application. On the right is an example of the Payment Object structure that will be returned to the merchant's application.

Once a Payment Object received, it can be used to create an Apple Pay payment transaction. To create an Apple Pay payment transaction you should specify the Encrypted Payment Token in the `<payment_token>` tag and the payment type you need in the `<payment_type>` tag. To specify Encrypted Payment Token use the value of the `token` field of the received Payment Object.

The following payment types are supported:

| Payment type | Description |
|---|---|
| authorize | behaves like common authorize transaction |
| recurring | behaves like common init_recurring_sale transaction |

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>apple_pay</transaction_type>
    <transaction_id>43671</transaction_id>
    <payment_type>authorize</payment_type>
    <payment_token>
      {
        "paymentData": { ... },
        "paymentMethod": { ... },
        "transactionIdentifier": "32B...4F3"
      }
    </payment_token>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>100</amount>
    <currency>USD</currency>
    <customer_email>travis@example.com</customer_email>
    <customer_phone>+1987987987987</customer_phone>
    <business_attributes>
      <event_start_date>16-08-2020</event_start_date>
      <event_end_date>25-08-2020</event_end_date>
      <event_organizer_id>20192375</event_organizer_id>
      <event_id>1912</event_id>
    </business_attributes>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>10178</zip_code>
      <city>Los Angeles</city>
      <state>CA</state>
      <country>US</country>
    </billing_address>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **apple_pay** |

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| payment_token | required | | Encrypted Payment Token |
| payment_type | required | string | Use either **authorize** for Authorize or **recurring** for Recurring transactions |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| customer_email | required* | e-mail address | Must contain valid e-mail of customer |
| customer_phone | required* | string(32) | Must contain valid phone number of customer |
| birth_date | required* | dd-mm-yyyy | Required when MCC is a Financial Services one (e.g. MCC 6012) and either card brand is Visa or Mastercard/Maestro with UK-based merchant, UK-based bin (domestic), and DEBIT card type |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| document_id | required* | sring(20) | Document ID of the consumer. See Document ID Parameter for more details. |
| **business_attributes** | required* | | Check business attributes section. |
| event_start_date | required* | dd-mm-yyyy | The date when event starts in format dd-mm-yyyy |
| event_end_date | required* | dd-mm-yyyy | The date when event ends in format dd-mm-yyyy |
| event_organizer_id | required* | string | |
| event_id | required* | string | |
| date_of_order | required* | dd-mm-yyyy | The date when order was placed in format dd-mm-yyyy |
| delivery_date | required* | dd-mm-yyyy | Date of the expected delivery in format dd-mm-yyyy |
| name_of_the_supplier | required* | string | |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required* | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

Successful Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```


```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>apple_pay</transaction_type>
  <status>approved</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <consumer_id>123456</consumer_id>
  <avs_response_code>5I</avs_response_code>
  <avs_response_text>Response provided by issuer processor; Address information not verified</avs_response_text>
  <authorization_code>345678</authorization_code>
  <response_code>00</response_code>
  <timestamp>2020-07-15T12:08:23Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
  <scheme_transaction_identifier>019091214161031</scheme_transaction_identifier>
</payment_transaction>
```


**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| consumer_id | string(10) | Consumer unique reference. See Consumers |
| gaming | string(255) | Signifies whether a gaming transaction is performed. Gaming transactions usually use MCC 7995, contact tech support for more details. |
| avs_response_code | string(255) | Generated by the card network on trying to match the billing address when performing the address verification. Optional, returned if config is enabled and acquirer supports it. Check AVS Status Codes for details. |
| avs_response_text | string(255) | Gives the human response text for the AVS response code above. Optional, returned if config is enabled and acquirer supports it. Check AVS Status Codes for details. |
| authorization_code | string(6) | Generated by the card network when an authorisation has occurred, used to identify that auth. Consists of 6 alphanumeric chars |
| response_code | string(2) | Defines the issuer result of a transaction, the status of a message or some action taken or required. See issuer response codes for details |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| redirect_url | url | URL where user has to be redirected to complete payment process. It is available for asynchronous mode |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08- 30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| partial_approval | string(4) | Optional, set to 'true' if partial approval happened. Partially approved amount is then in the amount field. Check Partial Approvals for details |
| sent_to_acquirer | string(255) | "true" or "false" |
| dynamic_descriptor_params | section | Optional, returned only if dynamic desc params are submitted on the API. Note here that the formatted dyn desc params are returned - as they would be submitted to the schemes for settlement. |
| scheme_transaction_identifier | string(32) | Id defined by card schemes |

Error Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>apple_pay</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>340</code>
  <technical_message>expiration_year is invalid</technical_message>
  <message>expiration_year is invalid</message>
  <timestamp>2020-07-15T12:08:23Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>false</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| gaming | string(255) | Signifies whether a gaming transaction is performed. Gaming transactions usually use MCC 7995, contact tech support for more details. |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08- 30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| dynamic_descriptor_params | section | Optional, returned only if dynamic desc params are submitted on the API. Note here that the formatted dyn desc params are returned - as they would be submitted to the schemes for settlement. |

#### AFRICAN MOBILE SALE

African Mobile Sale, otherwise known as Charge, is an APM used to process Mobile network operator payments. It is an async payment method and will be approved once the payment is processed with the Mobile network operator

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>african_mobile_sale</transaction_type>
   <transaction_id>43671</transaction_id>
   <usage>40208 concert tickets</usage>
   <remote_ip>245.253.2.12</remote_ip>
   <return_success_url>http://www.example.com/success</return_success_url>
   <return_failure_url>http://www.example.com/failure</return_failure_url>
   <amount>100</amount>
   <currency>KES</currency>
   <customer_email>barney.rubble@example.com</customer_email>
   <customer_phone>254701123456</customer_phone>
   <operator>SAFARICOM</operator>
   <target>000010</target>
   <billing_address>
     <first_name>Barney</first_name>
     <last_name>Rubble</last_name>
     <address1>14, Nerazdelni str</address1>
     <zip_code>1407</zip_code>
     <city>Nairobi</city>
     <country>KE</country>
   </billing_address>
   <risk_params>
     <user_id>123456</user_id>
   </risk_params>
 </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **african_mobile_sale** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| return_success_url | required | url | URL where customer is sent to after successful payment |
| return_failure_url | required | url | URL where customer is sent to after unsuccessful payment |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| operator | required | string(20) | Name of the Mobile network operator (MNO) which should process the transaction |
| target | required | string(20) | Number of the Paybill for which the transaction is intended |
| customer_email | required* | e-mail address | Must contain valid e-mail of customer |
| customer_phone | required | string(32) | Must contain valid phone number of customer |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |

| Parameter | Required | Format | Description |
|---|---|---|---|
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

**Supported countries, currencies, operators and payment method:**

| Country code | Currency code | Operator | Payment method |
|---|---|---|---|
| GH | GHS | VODACOM | M-PESA |
| KE | KES | SAFARICOM | M-PESA |
| UG | UGX | AIRTEL | Airtel Money |
| UG | UGX | MTN | MTN Mobile Money |

Successful Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>african_mobile_sale</transaction_type>
  <status>pending_async</status>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <timestamp>2020-07-15T12:08:23Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>KES</currency>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |

Error Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>african_mobile_sale</transaction_type>
  <status>error</status>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <message>Something went wrong, please contact support!</message>
  <technical_message>operator is not supported</technical_message>
  <timestamp>2020-07-15T12:08:23Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>KES</currency>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| message | string(255) | Human readable error message which can be displayed to users. |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |

---

# Reversals

Reversal transactions serve to change the state of the original transaction and return money back to customer's account. They can be used with Card, 3DS Card, and different APM transactions.

**REFUND**

Refunds allow to return already billed amounts to customers.

The amount can be fully or partially refunded. Refunds can only be done as follow transactions on former successfully processed transactions:

- Card transactions
- 3DS Card transactions
- Wallets
- Vouchers
- Online Banking ePayments
- Cash payments
- Gift Cards

Therefore, the reference id for the corresponding transaction is mandatory.

> This transaction type supports Level 3 travel data.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Refund');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setReferenceId('43672')
        ->setAmount('100')
        ->setCurrency('USD');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.RefundRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    RefundRequest request = new RefundRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setReferenceId("43672");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.refund(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "reference_id": "43672",
    "amount": "100",
    "currency": "USD"
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>refund</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <reference_id>43672</reference_id>
    <amount>100</amount>
    <currency>USD</currency>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **refund** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| reference_id | required | string(32) | Unique id returned by corresponding transaction |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| beneficiary_bank_code | required* | string | The bank code of the beneficiary's bank. |
| beneficiary_name | required* | string | The name of the beneficiary's bank. |
| beneficiary_account_number | required* | string | The account number of the beneficiary in his bank. |
| bank | optional | string | Name of the customer's bank |
| bank_branch | optional | string | Name of the Bank branch |
| bank_account | optional | string | Bank account number of the customer. |
| bank_account_type | optional | string(1) | The type of account. C: for Checking accounts, S: for Savings accounts, I: for International accounts |

`required* = conditionally required`

Beneficiary params will be required when refunding an online banking transaction with MYR currency. Contactech-support for more information.

Successful Response

```
stdClass Object
(
    [transaction_type] => refund
    [status] => approved
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [authorization_code] => 345678
    [retrieval_reference_number] => 016813015184
    [response_code] => 00
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:23.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
)
```

```
<payment_response content=[
    <transaction_type content=[refund]>
    <status content=[approved]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <authorization_code content=[345678]>
    <retrieval_reference_number content=[016813015184]>
    <response_code content=[00]>
    <timestamp content=[2020-07-15T12:08:23Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
]>
```

```
{
    transaction_type: "refund",
    status: "approved",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    authorization_code: "345678",
    retrieval_reference_number: "016813015184",
    response_code: "00",
    timestamp: "2020-07-15T12:08:23Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>refund</transaction_type>
  <status>approved</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <authorization_code>345678</authorization_code>
  <retrieval_reference_number>016813015184</retrieval_reference_number>
  <response_code>00</response_code>
  <timestamp>2020-07-15T12:08:23Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| authorization_code | string(6) | Generated by the card network when an authorisation has occurred, used to identify that auth. Consists of 6 alphanumeric chars |
| retrieval_reference_number | string(255) | A reference number used for tracking all messages related to a given cardholder transaction returned by some acquirers. |
| response_code | string(2) | Defines the issuer result of a transaction, the status of a message or some action taken or required. Seeissuer response codes for details |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08- 30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |

Error Response

```
stdClass Object
(
    [transaction_type] => refund
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 410
    [technical_message] => no approved reference transaction found
    [message] => no approved reference transaction found
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:23.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
)
```

```
<payment_response content=[
    <transaction_type content=[refund]>
    <status content=[error]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <code content=[410]>
    <technical_message content=[no approved reference transaction found]>
    <message content=[no approved reference transaction found]>
    <timestamp content=[2020-07-15T12:08:23Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
]>
```

```
{
    transaction_type: "refund",
    status: "error",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    code: "410",
    technical_message: "no approved reference transaction found",
    message: "no approved reference transaction found",
    timestamp: "2020-07-15T12:08:23Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>refund</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>410</code>
  <technical_message>no approved reference transaction found</technical_message>
  <message>no approved reference transaction found</message>
  <timestamp>2020-07-15T12:08:23Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| response_code | string(2) | Defines the issuer result of a transaction, the status of a message or some action taken or required. See issuer response codes for details |
| code | integer | Error code according to Error code table |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08- 30T17:46:11Z |

| Parameter | Type | Description |
|---|---|---|
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |

#### ASYNC REFUND

Async Refunds allow to return already billed amounts to customers where the transaction is confirmed asynchronously (i.e. the transaction is sent for processing without the client being blocked waiting and once the transaction is complete the client is notified about the result).

The amount can be fully or partially refunded. Async Refunds are only required for a few APMs. Please contact tech-support@e-comprocessing.com for more details.

Similarly to ordinary Refunds, the reference id for the corresponding transaction is mandatory.

> This transaction type supports Level 3 travel data.

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>async_refund</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <reference_id>43672</reference_id>
    <amount>100</amount>
    <currency>USD</currency>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **async_refund** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| reference_id | required | string(32) | Unique id returned by corresponding transaction |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| beneficiary_bank_code | required* | string | The bank code of the beneficiary's bank. |
| beneficiary_name | required* | string | The name of the beneficiary's bank. |
| beneficiary_account_number | required* | string | The account number of the beneficiary in his bank. |
| bank | optional | string | Name of the customer's bank |
| bank_branch | optional | string | Name of the Bank branch |
| bank_account | optional | string | Bank account number of the customer. |
| bank_account_type | optional | string(1) | The type of account. C: for Checking accounts, S: for Savings accounts, I: for International accounts |

`required* = conditionally required`

Successful Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>async_refund</transaction_type>
  <status>approved</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <authorization_code>345678</authorization_code>
  <response_code>00</response_code>
  <timestamp>2020-07-15T12:08:23Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| authorization_code | string(6) | Generated by the card network when an authorisation has occurred, used to identify that auth. Consists of 6 alphanumeric chars |
| response_code | string(2) | Defines the issuer result of a transaction, the status of a message or some action taken or required. Seeissuer response codes for details |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08- 30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |

Error Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>async_refund</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>410</code>
  <technical_message>no approved reference transaction found</technical_message>
  <message>no approved reference transaction found</message>
  <timestamp>2020-07-15T12:08:23Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| transaction_id | string(255) | Unique transaction id defined by merchant |

| Parameter | Type | Description |
|---|---|---|
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| response_code | string(2) | Defines the issuer result of a transaction, the status of a message or some action taken or required. See issuer response codes for details |
| code | integer | Error code according to Error code table |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08- 30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |

**VOID**

Void transactions undo other transactions.

Transactions of types authorize, capture, sale, refund, recurring sale, and init recurring sale along with their 3D variants can be reversed on the same day the transaction took place. The transaction will not show up on the customer's credit card statement if voided on the same day.

> Not captured authorize and authorize3d transactions can be voided without a specific timeframe.

> The same day is dependent of the timezone of the acquiring bank.

> This transaction can also be used to fully reverse a Preauthorization The void time-frame in this case depends on the preauthorization specifics (Cardbrand, Merchant Category Code etc). To learn more about this, navigate to the Full Reversal section.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cancel');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setReferenceId('43672');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.VoidRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    VoidRequest request = new VoidRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setReferenceId("43672");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.cancel(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "reference_id": "43672"
  }
).send()
  .then(success)
  .catch(failure);
```

```bash
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>void</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <reference_id>43672</reference_id>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **void** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| reference_id | required | string(32) | Unique id returned by corresponding transaction |

`required* = conditionally required`

Successful Response

```
stdClass Object
(
    [transaction_type] => void
    [status] => approved
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [authorization_code] => 345678
    [retrieval_reference_number] => 016813015184
    [response_code] => 00
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:23.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[void]>
    <status content=[approved]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <authorization_code content=[345678]>
    <retrieval_reference_number content=[016813015184]>
    <response_code content=[00]>
    <timestamp content=[2020-07-15T12:08:23Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[true]>
]>
```

```
{
    transaction_type: "void",
    status: "approved",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    authorization_code: "345678",
    retrieval_reference_number: "016813015184",
    response_code: "00",
    timestamp: "2020-07-15T12:08:23Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "true",
}
```

```
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>void</transaction_type>
    <status>approved</status>
    <mode>live</mode>
    <transaction_id>43671</transaction_id>
    <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
    <authorization_code>345678</authorization_code>
    <retrieval_reference_number>016813015184</retrieval_reference_number>
    <response_code>00</response_code>
    <timestamp>2020-07-15T12:08:23Z</timestamp>
    <descriptor>Descriptor one</descriptor>
    <amount>100</amount>
    <currency>USD</currency>
    <sent_to_acquirer>true</sent_to_acquirer>
  </payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| authorization_code | string(6) | Generated by the card network when an authorisation has occurred, used to identify that auth. Consists of 6 alphanumeric chars |
| retrieval_reference_number | string(255) | A reference number used for tracking all messages related to a given cardholder transaction returned by some acquirers. |

| Parameter | Type | Description |
|---|---|---|
| response_code | string(2) | Defines the issuer result of a transaction, the status of a message or some action taken or required. Seeissuer response codes for details |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08- 30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [transaction_type] => void
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 420
    [technical_message] => can not do void on void reference
    [message] => can not do void on void reference
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:23.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
    [sent_to_acquirer] => false
)
```

```
<payment_response content=[
    <transaction_type content=[void]>
    <status content=[error]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <code content=[420]>
    <technical_message content=[can not do void on void reference]>
    <message content=[can not do void on void reference]>
    <timestamp content=[2020-07-15T12:08:23Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[100]>
    <currency content=[USD]>
    <sent_to_acquirer content=[false]>
]>
```

```
{
    transaction_type: "void",
    status: "error",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    code: "420",
    technical_message: "can not do void on void reference",
    message: "can not do void on void reference",
    timestamp: "2020-07-15T12:08:23Z",
    descriptor: "Descriptor one",
    amount: "100",
    currency: "USD",
    sent_to_acquirer: "false",
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>void</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>420</code>
  <technical_message>can not do void on void reference</technical_message>
  <message>can not do void on void reference</message>
  <timestamp>2020-07-15T12:08:23Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
  <sent_to_acquirer>false</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| response_code | string(2) | Defines the issuer result of a transaction, the status of a message or some action taken or required. See issuer response codes for details |
| code | integer | Error code according to Error code table |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08- 30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

#### KLARNA REFUND

**Klarna Refunds allow to return already billed amounts to customers.**

The amount can be fully or partially refunded. Klarna refunds can only be done on former Klarna Capture (settled) transactions.

Therefore, the `reference_id` for the corresponding transaction is mandatory.

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```xml
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>klarna_refund</transaction_type>
   <transaction_id>43671</transaction_id>
   <usage>40208 concert tickets</usage>
   <remote_ip>245.253.2.12</remote_ip>
   <reference_id>43672</reference_id>
   <amount>60</amount>
   <currency>USD</currency>
   <items>
     <item>
       <item_type>physical</item_type>
       <reference>19-402-USA</reference>
       <name>BatteryPowerPack</name>
       <quantity>1</quantity>
       <unit_price>60</unit_price>
       <tax_rate>0</tax_rate>
       <total_amount>60</total_amount>
       <total_discount_amount>0</total_discount_amount>
       <total_tax_amount>0</total_tax_amount>
       <image_url>https://example.com/image_url</image_url>
       <product_url>https://example.com/product_url</product_url>
       <quantity_unit>pcs</quantity_unit>
       <merchant_data>
         <marketplace_seller_info>Electronic gadgets</marketplace_seller_info>
       </merchant_data>
     </item>
   </items>
 </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **klarna_refund** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| reference_id | required | string(32) | Unique id returned by corresponding transaction |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |
| **items** | required | | List with items |
| item_type | required | string(255) | Order line type. Possible values: Supported item types |
| quantity | required | integer | Non-negative. The item quantity |
| unit_price | required | integer | Minor units. Includes tax, excludes discount(max value: 100000000) |
| total_amount | required | integer | Includes tax and discount. Must match (quantity unit price) - total discount amount divided by quantity (max value: 100000000) |
| reference | optional | string(255) | Article number, SKU or similar |
| name | optional | string(255) | Descriptive item name |
| tax_rate | optional | integer | Non-negative. In percent, two implicit decimals. I.e 2500 = 25.00 percent |
| total_discount_amount | optional | integer | Non-negative minor units. Includes tax |
| total_tax_amount | optional | integer | Must be within 1 of total amount - total_amount * 10000 / (10000 + tax rate). Negative when type is discount |
| image_url | optional | url | URL to an image that can be later embedded in communications between Klarna and the customer. (max 1024 characters) |
| product_url | optional | url | URL to an image that can be later embedded in communications between Klarna and the customer. (max 1024 characters) |
| quantity_unit | optional | string(8) | Unit used to describe the quantity, e.g. kg, pcs... If defined has to be 1-8 characters |
| **product_identifiers** | optional | | List with product identifiers |
| brand | optional | string(255) | The product's brand name as generally recognized by consumers. If no brand is available for a product, do not supply any value |
| category_path | optional | string(255) | The product's category path as used in the merchant's webshop. Include the full and most detailed category and separate the segments with ' > ' |

| Parameter | Required | Format | Description |
|---|---|---|---|
| global_trade_item_number | optional | string(255) | The product's Global Trade Item Number (GTIN). Common types of GTIN are EAN, ISBN or UPC. Exclude dashes and spaces, where possible |
| manufacturer_part_number | optional | string(255) | The product's Manufacturer Part Number (MPN), which - together with the brand - uniquely identifies a product. Only submit MPNs assigned by a manufacturer and use the most specific MPN possible |
| **merchant_data** | optional | | List with merchant data |
| marketplace_seller_info | optional | string(255) | Information for merchant marketplace |

`required* = conditionally required`

Supported item types:

| Item Types |
|---|
| physical |
| discount |
| shipping_fee |
| sales_tax |
| digital |
| gift_card |
| store_credit |
| surcharge |

Successful Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>klarna_refund</transaction_type>
  <status>approved</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <timestamp>2020-07-15T12:08:23Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>60</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |

| Parameter | Type | Description |
|---|---|---|
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>klarna_refund</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>420</code>
  <technical_message>can not do void on void reference</technical_message>
  <message>can not do void on void reference</message>
  <timestamp>2020-07-15T12:08:23Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>60</amount>
  <currency>USD</currency>
  <sent_to_acquirer>false</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

#### BITPAY REFUND

BitPay Refund is a custom refund method which will handle the asynchronous BitPay refund workflow. BitPay refunds can only be done on former transactions. Therefore, the reference id for the corresponding BitPay Sale transaction is mandatory.

BitPay Refund is an asynchronous transaction type.

When a BitPay Refund is requested, BitPay will send an email to the consumer with a request to provide the refund crypto address. This request will be valid for 3 days and will expire afterwards. When the crypto address is provided, the refund will be processed (processing usually takes 24 hours).

A Notification will be sent to the Merchant when the Bitpay refund is completed.

> Only full refunds are supported at the moment.

> BitPay Refunds can be voided only in the 24-hour processing period.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Crypto\BitPay\Refund');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setReferenceId('43672')
        ->setAmount('100')
        ->setCurrency('USD');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```curl
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>bitpay_refund</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <reference_id>43672</reference_id>
    <amount>100</amount>
    <currency>USD</currency>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **bitpay_refund** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| reference_id | required | string(32) | The reference_id must be a BitPay Sale transaction |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | required | string(3) | Currency code in ISO 4217 |

`required* = conditionally required`

Successful Response

```
stdClass Object
(
    [transaction_type] => bitpay_refund
    [status] => approved
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:23.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>bitpay_refund</transaction_type>
  <status>approved</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <timestamp>2020-07-15T12:08:23Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
</payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |

Error Response

```
stdClass Object
(
    [transaction_type] => bitpay_refund
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 410
    [technical_message] => no approved reference transaction found
    [message] => no approved reference transaction found
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:23.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 100
    [currency] => USD
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>bitpay_refund</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>410</code>
  <technical_message>no approved reference transaction found</technical_message>
  <message>no approved reference transaction found</message>
  <timestamp>2020-07-15T12:08:23Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>100</amount>
  <currency>USD</currency>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| response_code | string(2) | Defines the issuer result of a transaction, the status of a message or some action taken or required. See issuer response codes for details |
| code | integer | Error code according to Error code table |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08- 30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |

#### PARTIAL REVERSAL

Partial reversal transactions are used in the preauthorization workflow to release a part of the total authorized amount.

> For more information, navigate to the Preauthorization Partial Reversal section in the preauthorization workflow.

---

## PayByLink

PayByLink is a frictionless payment via link. It provides merchants the ability to send a payment link to a customer via email or SMS (configurable), using Virtual Terminal.

#### WORKFLOW

The merchant can generate a PayByLink payment from the Virtual Terminal (if feature is enabled) by customizing an email template for sending payment link or configuring preferred medium (email/SMS) for sending payment link. When all the needed fields are filled in and the mandatory initial payload is provided, a payment request to the WPF is initiated. As a result of this request (if successful) the response would include redirect URL, which would be sent either via email or SMS to the customer (channel and needed email/phone number should be provided by the merchant when initiating the PayByLink payment). The customer can complete the payment by following the provided URL. It redirects to the merchant's web payment form (WPF) where the redirection workflows are the same as the ones described in the WPF section. The WPF reconcile API could also be used to check the status of any PayByLink initiated payment.

#### COMBINATION WITH PAY LATER FUNCTIONALITY

The PayByLink triggered payments could be easily combined with the 'Pay Later' functionality (available also for the WPF API). The PayByLink form provides the ability to choose whether the customer would have the option enabled to delay the payment and complete it later. It also gives the ability of enqueuing reminders based on pre-configured values. The reminders include the URL for payment completion as well.

#### REMINDERS CONFIGURATION

- Up to 3 reminders can be configured for each payment.
- The available channels for sending reminders are `email` and `sms`.
- The time for sending a reminder is set in number of minutes after payment creation.
- The time for sending of each reminder shouldn't be greater that the configured payment lifetime.

For configuration options or any other additional questions you can always contact Tech Support at tech-support@e-comprocessing.com.

# Alternative Payment Method External Events

#### INTRODUCTION

For some alternative payment method (APM) transactions additional events may occur resulting from various actions from the part of the consumer or the merchant. Examples: returned/reversed bank transfers, funds not received, additional bank transfers made using the same transaction reference number etc.

In Genesis these are called external events and are handled in the following manner:

1. A transaction note is created for the external event (visible under the ”Transaction Notes” section on the corresponding payment transaction page in the merchant console).
2. An API notification is sent to the merchant notification endpoint
3. An email notification is sent to the merchant admin email address
4. The original transaction status might be updated depending on the nature of the external event

#### LIST OF EXTERNAL EVENTS PER ALTERNATIVE PAYMENT METHOD

| APM | External event | Description | Status change |
|---|---|---|---|
| InstaDebit Payin | instadebit_payin_return | Payment has been returned to the consumer | voided |
| InstaDebit Payin | instadebit_payin_adjustment | Payment has been adjusted | none |
| InstaDebit Payout | instadebit_payout_return | Payment has been returned to the consumer | voided |
| InstaDebit Payout | instadebit_payout_adjustment | Payment has been adjusted | none |
| iDebit Payin | idebit_payin_return | Payment has been returned to the consumer | voided |
| iDebit Payin | idebit_payin_adjustment | Payment has been adjusted | none |
| iDebit Payout | idebit_payout_return | Payment has been returned to the consumer | voided |
| iDebit Payout | idebit_payout_adjustment | Payment has been adjusted | none |
| P24 | p24_external_refund | Payment has been rescinded by the consumer or was never received | refunded |
| Argencard | chargeback_external_event | Payment has been considered fraudulent by the card issuer and was reversed | chargebacked |
| Aura | chargeback_external_event | Payment has been considered fraudulent by the card issuer and was reversed | chargebacked |
| Cabal | chargeback_external_event | Payment has been considered fraudulent by the card issuer and was reversed | chargebacked |
| Cencosud | chargeback_external_event | Payment has been considered fraudulent by the card issuer and was reversed | chargebacked |
| Elo | chargeback_external_event | Payment has been considered fraudulent by the card issuer and was reversed | chargebacked |
| Naranja | chargeback_external_event | Payment has been considered fraudulent by the card issuer and was reversed | chargebacked |
| Nativa | chargeback_external_event | Payment has been considered fraudulent by the card issuer and was reversed | chargebacked |
| Tarjeta Shopping | chargeback_external_event | Payment has been considered fraudulent by the card issuer and was reversed | chargebacked |

#### EMAIL NOTIFICATION

An email is sent to the merchant admin’s email address detailing the external event type and any other relevant details together with a link to the original payment transaction.

#### API NOTIFICATION

Merchants will receive API notifications every time Genesis obtains information about APM external events. Notifications are transmitted via HTTP POST (application/x-www-form-urlencoded) to the notification url endpoint provided in the XML request or to the Notification URL from the merchant account

An example notification:

```
&notification_type=apm_external_event
&signature=8216b1b4929c0a41ed44bf51726c10872a78f181b26d1427e15410da56803fc0a2f
&payment_transaction_unique_id=64216236bced683952325b6698b3954a
&category=citadel_payin_chargeback
&priority=info
&code=SDR
&info=Revoked+payment+due+to+instant+payment+funds+not+received
&message=A+Citadel+Payin+transaction+has+been+reversed%2Frevoked.+Payment+has+been+rescinded+by+the+consumer+or+funds+were+never+received+for+the+payment.
&payload=...
```

**Parameters:**

| Name | Type | Description |
|---|---|---|
| notification_type | string | constant value ”apm external event” |

| Name | Type | Description |
|---|---|---|
| signature | string | the signature of the notification, should be used to verify the the notification was sent by Genesis |
| payment_transaction_unique_id | string | unique id of the original transaction, generated by Genesis |
| category | string | type of the external event |
| priority | string | can be one of ”info”, ”normal” or ”urgent” |
| code | string | code for the external event from the APM provider system |
| info | string | a short description of the external event |
| message | string | full short description of the external event |
| payload | string | the raw response for the external event as received from the APM provider |

The signature is a security measure meant to ensure that the gateway is really the sender of the notification. It is generated by concatenating the unique id of the payment with your API password and generating a SHA-512 Hash (Hex) of the string:

```
SHA-512 Hash Hex of [payment_transaction_unique_id][Your Merchant API password]
```

**Notification signature examples**

| payment_transaction_unique_id | API password | signature |
|---|---|---|
| 26aa150ee68b1b2d6758a0e6c44fce4c | 50fd87e65eb415f42fb5af4c9cf497662e00b785 | c5219b3d385e74496b2b48a549 |
| 3f760162ef57a829011e5e2379b3fa17 | 50fd87e65eb415f42fb5af4c9cf497662e00b785 | 14519d0db2f7f8f407efccc9b09 |

```
<?xml version="1.0" encoding="UTF-8"?>
<notification_echo>
  <unique_id>3f760162ef57a829011e5e2379b3fa17</unique_id>
</notification_echo>
```

When receiving the notification, you are required to render an XML page containing the transaction’s payment transaction unique id so that the gateway knows that you have accepted the notification. If the XML is not delivered, the notification is sent periodically as per the rules for notifications delivery.

---

# Advanced risk management with RiskParams

The risk params section in the payment transaction xml allows you to pass user specific values along with the payment transaction. These values may be used by advanced risk management features and checked against a blacklist.

RiskParams can be used in any user triggered payment transaction. User triggered transactions types areAuthorize, Authorize3d, Sale, Sale3d, InitRecurringSale, InitRecurringSale3d, and AccountVerification.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Sale');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('4200000000000000')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')


        // Risk Params
        ->setRiskSsn('987-65-4320')
        ->setRiskMacAddress('12-34-56-78-9A-BC')
        ->setRiskSessionId('1DA53551-5C60-498C-9C18-8456BDBA74A9')
        ->setRiskUserId('1002547')
        ->setRiskUserLevel('vip')
        ->setRiskEmail('test@example.com')
        ->setRiskPhone('+49301234567')
        ->setRiskRemoteIp('245.253.2.12');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.SaleRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    SaleRequest request = new SaleRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("4200000000000000");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");


    // Risk Params
    request.setRiskSsn("987-65-4320");
    request.setRiskMacAddress("12-34-56-78-9A-BC");
    request.setRiskSessionId("1DA53551-5C60-498C-9C18-8456BDBA74A9");
    request.setRiskUserId("1002547");
    request.setRiskUserLevel("vip");
    request.setRiskEmail("test@example.com");
    request.setRiskPhone("+49301234567");
    request.setRiskRemoteIp("245.253.2.12");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.sale(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "4200000000000000",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    },
    "risk_params": {
      "ssn": "987-65-4320",
      "mac_address": "12-34-56-78-9A-BC",
      "session_id": "1DA53551-5C60-498C-9C18-8456BDBA74A9",
      "user_id": "1002547",
      "user_level": "vip",
      "email": "test@example.com",
      "phone": "+49301234567",
      "remote_ip": "245.253.2.12"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>sale</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>100</amount>
    <currency>USD</currency>
    <card_holder>Travis Pastrana</card_holder>
    <card_number>4200000000000000</card_number>
    <expiration_month>12</expiration_month>
    <expiration_year>2021</expiration_year>
    <cvv>834</cvv>
    <customer_email>travis@example.com</customer_email>
    <customer_phone>+1987987987987</customer_phone>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>10178</zip_code>
      <city>Los Angeles</city>
      <state>CA</state>
      <country>US</country>
    </billing_address>
    <risk_params>
      <ssn>987-65-4320</ssn>
      <mac_address>12-34-56-78-9A-BC</mac_address>
      <session_id>1DA53551-5C60-498C-9C18-8456BDBA74A9</session_id>
      <user_id>1002547</user_id>
      <user_level>vip</user_level>
      <email>test@example.com</email>
      <phone>+49301234567</phone>
      <remote_ip>245.253.2.12</remote_ip>
    </risk_params>
  </payment_transaction>'
```

### Request Parameters

| Parameter | Required | Format | Description |
|---|---|---|---|

| Parameter | Required | Format | Description |
|---|---|---|---|
| **risk_params** | optional | | |
| ssn | optional | string(128) | Social Security number or equivalent value for non US customers. |
| mac_address | optional | string(128) | The customers mac address. |
| session_id | optional | string(128) | The customers session_id. |
| user_id | optional | string(128) | The customers user_id. |
| user_level | optional | string(128) | A value describing the customers trust level, may be used by the risk management for configurable differentiated limits. |
| email | optional | string(128) | The customers email. |
| phone | optional | string(128) | The customers phone. |
| remote_ip | optional | string(128) | The customers ip address. |
| serial_number | optional | string(128) | Custom serial number. |
| pan_tail | optional | string(128) | The last 4 digits of the card number. |
| bin | optional | string(128) | The first 6 digits of the card number. |
| first_name | optional | string(128) | Customer first name. |
| last_name | optional | string(128) | Customer last name. |
| country | optional | string(128) | The country of the customer. |
| pan | optional | string(128) | The PAN hash of the customer card number. |
| forwarded_ip | optional | string(128) | MaxMind specific risk param. |
| username | optional | string(128) | MaxMind specific risk param. |
| password | optional | string(128) | MaxMind specific risk param. |
| bin_name | optional | string(128) | MaxMind specific risk param. |
| bin_phone | optional | string(128) | MaxMind specific risk param. |

`required* = conditionally required`

> The risk management needs to be configured to use these values, passing the values alone will not trigger any risk management features.

To use these values for risk management please contactour Risk team.

---

## Credential On File (COF)

As the payment ecosystem has evolved, instances in which a transaction is initiated with a stored credential based on cardholder consent for future use have significantly increased. Growth in digital commerce, together with the emergence of new business models, has increased the number of transactions where a merchant or its agent, a payment facilitator (PF), or a staged digital wallet operator (SDWO) uses cardholder payment credentials (i.e., account details) that they previously stored for future purchases.

In Genesis, the COF indicator can be used for the following transaction types:Account Verification, Authorize, Authorize3D, Sale, Sale3D, InitRecurringSale, InitRecurringSale3D, Payout to mark a transaction as **initial customer initiated, subsequent customer initiated or as unscheduled merchant initiated (UCOF)**.

The UCOF transaction uses a previously stored credential for a fixed or variable amount and it does not occur on a scheduled or regularly occurring transaction date. With it, the cardholder has provided consent to the merchant to initiate one or more future transactions. An example of such a transaction is an account auto-top up.

**Supported options for Credential On File (COF) field:**

| COF | Description |
|---|---|
| initial_customer_initiated | Initial transaction used to store payment credentials for future customer initiated payments while processing. Required for external tokenization, and optional for gateway-based tokenization |
| subsequent_customer_initiated | Subsequent customer initiated transaction using previously stored payment credentials. Required for external tokenization, and optional for gateway-based tokenization |
| merchant_unscheduled | Merchant initiated transaction where the customer has given prior consent for the merchant to store and use payment credentials, UCOF transaction. For UCOF transaction in the request should be sent scheme transaction identifier of the initial transaction, used to store customer payment data. |

---

## Currency and Amount Handling

The gateway handles all types of processing currencies, with exponents ranging from 0 (e.g. JPY), 2 (e.g. CNY, USD, EUR, GBP), to 3 (e.g. KWD). Processing currencies are configured on

terminal level.

Transaction amounts on the API level should be submitted in the minor currency unit for the given currency, e.g.:

**Amount currencies:**

| Name | Type | Description |
|---|---|---|
| USD | 100.33 | Should be submitted as 10033 in the amount API field (exponent 2) |
| EUR | 3 | Should be submitted as 300 in the amount API field (exponent 2) |
| JPY | 150 | Should be submitted as 150 in the amount API field (exponent 0) |
| KWD | 100.333 | Should be submitted as 100333 in the amount API field (exponent 3) |

**Amount limits:** Amount has to be provided within limit for listed transaction types and currencies:

| Transaction Type | Currency | minimum | maximum |
|---|---|---|---|
| Global limit | All | 0.01 | 1,000,000,000.00 |
| alipay | CNY | 0.01 | 50,000.00 |
| | EUR | 0.01 | 6,529.00 |
| qiwi | EUR | 5.00 | 350.00 |
| | USD | 5.00 | 500.00 |
| | RUB | 500.00 | 15,000.00 |
| | KZT | 500.00 | 74,300.00 |
| davivienda | USD | 0.01 | 3,000.00 |
| banco de chile | USD | 0.01 | 3,000.00 |
| webpay | USD | 0.01 | 3,000.00 |
| pago facil | USD | 0.01 | 3,000.00 |
| rapi pago | USD | 0.01 | 3,000.00 |
| link | USD | 0.01 | 3,000.00 |
| santander | USD | 0.01 | 3,000.00 |
| aura | USD | 0.01 | 3,000.00 |
| cabal | USD | 0.01 | 3,000.00 |
| nativa | USD | 0.01 | 3,000.00 |
| naranja | USD | 0.01 | 3,000.00 |
| cencosud | USD | 0.01 | 3,000.00 |
| tarjeta shopping | USD | 0.01 | 3,000.00 |
| redpagos | USD | 0.01 | 3,000.00 |
| bcmc | EUR | 1,00 | 1,000,000.00 |
| elo | USD | 0.01 | 3,000.00 |
| oxxo | USD | 0.01 | 3,000.00 |
| bradesco | USD | 0.01 | 3,000.00 |
| cartao mercado livre | USD | 0.01 | 3,000.00 |
| efecty | USD | 0.01 | 3,000.00 |
| boleto | USD | 2.50 | 2,500.00 |
| itau | USD | 0.01 | 3,000.00 |
| multibanco | USD | 0.01 | 99,999.99 |
| banco do brasil | USD | 0.01 | 3,000.00 |
| argencard | USD | 0.01 | 3,000.00 |
| banco de occidente | USD | 0.01 | 3,000.00 |
| bancomer | USD | 0.01 | 3,000.00 |

| Transaction Type | Currency | minimum | maximum |
|---|---|---|---|
| giropay | EUR | 1.00 | 1,000,000,000.00 |
| baloto | USD | 0.01 | 3,000.00 |
| eps | EUR | 1.00 | 1,000,000.00 |
| sofort | EUR | 1.00 | 5,000.00 |
| sdd sale | All | 0.10 | 24,999.99 |
| sct payout | All | 0.10 | 24,999.99 |
| sdd init recurring sale | All | 0.10 | 24,999.99 |
| sdd refund | All | 0.10 | 24,999.99 |
| neosurf | All | 0.01 | 9,999.99 |
| p24 | EUR | 0.01 | 15,000.00 |
| rpn payment | All | 0.10 | 100,000.00 |
| rpn payout | All | 100.00 | 1,000,000.00 |
| citadel payin | EUR | 0.01 | 10,000.00 |
| citadel payout | EUR | 0.01 | 10,000.00 |
| idebit payin | CAD | 0.01 | 1,500.00 |
| idebit payout | CAD | 0.01 | 1,500.00 |
| online banking | CNY* | 10.00 | 50,000.00 |
| | THB | 10.00 | 500,000.00 |
| | IDR | 10,000.00 | 50,000,000.00 |
| | MYR | 10.00 | 20,000.00 |
| bank payout | CNY | 60.00 | 49,000.00 |
| | THB | 350.00 | 175,000.00 |
| | IDR | 50,000.00 | 25,000,000.00 |
| | MYR | 50.00 | 20,000.00 |
| wechat | All | 10.00 | 3,000.00 |
| ezeecard payout | EUR | 10.00 | 800.00 |
| paysafecard | EUR | 0.01 | 1,000.00 |
| poli | AUD | 0.01 | 9,999.00 |
| insta debit payin | CAD | 0.01 | 1,500.00 |
| bitpay | USD | 1.00 | 950,000.00 |
| bitpay sale | USD | 1.00 | 950,000.00 |
| bitpay payout | USD | 1.00 | 950,000.00 |
| pse | USD | 0.01 | 3,000.00 |

`* = Depends on the setup`

Check the ISO 4217 standard for details on currencies and their exponents/minor currency units.

---

## Dynamic Descriptor

Dynamic descriptor functionality is available as part of the gateway. It is enabled on terminal level, so contact the IT Support team attech-support@e-comprocessing.com if you wish to use it.

Currently, the transactions types that support dynamic descriptor parameters are Authorize, Authorize3d, Sale, Sale3d, InitRecurringSale, InitRecurringSale3d and Payout.

WPF payments also support dynamic descriptor.

The currently supported dynamic descriptor parameters are:

| Name | Type | Description |
|---|---|---|

| Name | Type | Description |
|---|---|---|
| merchant_name | string(25) | Needed by merchants/PSPs to change the charge description. Any chars above 25 are truncated. If the merchant name is less than 25 chars, it is right padded with whitespace before sent to the schemes |
| merchant_city | string(13) | Contains the city of the merchant for card-present merchants and the merchant phone number for CNP merchants. Formatting rules applied as mentioned above |
| sub_merchant_id | string(15) | Sub-merchant ID assigned by the Payment Facilitator. |

If the terminal is configured with dynamic descriptor, but the merchant does not send the dynamic descriptor parameter in question at all or sends whitespace, this defaults the given dynamic descriptor parameter to the static descriptor configured on MID level when submitted to the schemes.

Note also that the dynamic descriptor params section with the properly formatted individual dynamic descriptor params is returned in the payment transaction response on the API if those params have been submitted in the transaction API request beforehand.

Regarding reference-based transactions, since they do not accept any dynamic descriptor params in the request, there is no dynamic descriptor params section in the response as well. Reference-based transactions reuse the dynamic descriptor params of their original transaction.

The sub-merchant ID should be provided by all merchants that are processing under Payment Facilitator. If the terminal does not support dynamic descriptor, a static value can be configured on a merchant level.

---

## Electronic Commerce Indicator

Electronic Commerce Indicator (ECI) is a value that is returned from the Directory Server (Visa, MasterCard, etc.) to indicate the authentication results of your customer's credit card payment on 3D Secure.

Visa/JCB/Diners/American Express/Rupay

| ECI Code | Description |
|---|---|
| 05 | Both cardholder and card issuing bank are 3D enabled. 3D card authentication is successful |
| 06 | Either cardholder or card issuing bank is not 3D enrolled. 3D card authentication is unsuccessful, in sample situations as: 1. 3D cardholder not enrolled 2. Card issuing bank is not 3D Secure ready |
| 07 | Authentication is unsuccessful or not attempted. The credit card is either a non-3D card or card issuing bank does not handle it as a 3D transaction |

MasterCard/Maestro

| ECI Code | Description |
|---|---|
| 02 | Both cardholder and card issuing bank are 3D enabled. 3D card authentication is successful |
| 01 | Either cardholder or card issuing bank is not 3D enrolled. 3D card authentication is unsuccessful, in sample situations as: 1. 3D Cardholder not enrolled 2. Card issuing bank is not 3D Secure ready |
| 00 (or empty) | Authentication is unsuccessful or not attempted. The credit card is either a non-3D card or card issuing bank does not handle it as a 3D transaction |

---

## Issuer Response Codes

See below a list of issuer response codes with the corresponding messages. Issuer response codes (response code element) are different than the regular gateway codes (code element) - the issuer response code maps to the issuer code while the code is the gateway internal code mapping and part of the API, as described in the Errors section. Transaction responses will return the relevant issuer response code (note that both issuer response code and authorization code are optional in the API responses and will be returned only if the transaction reached the issuer)

| Issuer Response Code | Issuer Message |
|---|---|
| 00 | Approved or completed successfully |
| 02 | Refer to card issuer |
| 03 | Invalid merchant |
| 04 | Pickup card |
| 05 | Do not honour |
| 06 | Invalid Transaction for Terminal |
| 07 | Honour with ID |
| 08 | Time-Out |
| 09 | No Original |
| 10 | Unable to Reverse |

| Issuer Response Code | Issuer Message |
|---|---|
| 11 | Partial Approval |
| 12 | Invalid transaction card / issuer / acquirer |
| 13 | Invalid amount |
| 14 | Invalid card number |
| 17 | Invalid Capture date, terminal business date |
| 19 | System Error, Re-enter transaction |
| 20 | No From Account |
| 21 | No To Account |
| 22 | No Checking Account |
| 23 | No Saving Account |
| 24 | No Credit Account |
| 30 | Format error |
| 34 | Implausible card data |
| 39 | Transaction Not Allowed |
| 41 | Pick-up card |
| 42 | Special Pickup |
| 43 | Hot Card, Pickup if possible |
| 44 | Pickup Card |
| 45 | Transaction Back Off |
| 51 | Not su cient funds |
| 54 | Expired card |
| 55 | Incorrect PIN, Re-enter |
| 57 | Not permitted on card |
| 58 | Txn Not Permitted On Term |
| 61 | Exceeds amount limit |
| 62 | Restricted card |
| 63 | MAC Key Error |
| 65 | Exceeds frequency limit |
| 66 | Exceeds Acquirer Limit |
| 67 | Retain Card, no reason specified |
| 68 | Response received too late |
| 75 | Exceeds PIN Retry |
| 76 | Invalid Account |
| 77 | Issuer Does Not Participate In The Service |
| 78 | Function Not Available |
| 79 | Key Validation Error |
| 80 | Approval for Purchase Amount Only |
| 81 | Unable to Verify PIN |
| 82 | Invalid Card Verification Value |
| 83 | Not declined, AVS Only |
| 84 | Invalid Life Cycle of transaction |
| 85 | No Keys To Use |

| Issuer Response Code | Issuer Message |
|---|---|
| 86 | K M E Sync Error |
| 87 | PIN Key Error |
| 88 | MAC sync Error |
| 89 | Security Violation |
| 91 | Issuer not available |
| 92 | Invalid Issuer |
| 93 | Transaction cannot be completed |
| 94 | Invalid originator |
| 96 | System malfunction |
| 97 | No Funds Transfer |
| 98 | Duplicate Reversal |
| 99 | Duplicate Transaction |
| N3 | Cash Service Not Available |
| N4 | Cash Back Request Exceeds Issuer Limit |
| N7 | CVV2 Failure |
| R0 | Stop Payment Order |
| R1 | Revocation of Authorisation Order |
| R3 | Revocation of all Authorisations Order |

## Manually Reviewed Transactions

Under certain conditions, transactions can be stopped for manual review by the Risk team. This happens when the appropriate risk rules have been enabled for the merchant in question. Feel free to discuss enabling of manual reviews for transactions with our Risk team. Transactions that are stopped for manual review are returned with status 'pending review' in the API response, together with a detailed message specifying a manual review of this transaction. Note that transactions will be manually reviewed by the Risk team in the next 24 hours. In the process, each transaction will be manually approved or manually declined, and at this point the merchant will receive a notification with the status of the transaction, see Notifications.

Special case for manual reviewing is when the transaction is 3D async, in this case the merchant receives one notification if the transaction is manually declined by the Risk team, and two notifications if the transaction is manually approved. On manual approval, the first notification is sent once the transaction is manually approved and is sent for enrollment check to the MPI provider - the notification will contain status pending asyc together with the redirect url where the consumer needs to be redirected to by the merchant. The second notification is the standard 3D notification, once a consumer has been redirected to the given redirect url, has entered his/her MPI password, and contains the final status of the transaction whether it is approved by the issuer, or declined for invalid 3D password, and so on

The format of the first notification for manual review and following approval is:

```
?transaction_id=82803B4C-70CC-43BD-8B21-FD0395285B40
&unique_id=44177a21403427eb96664a6d7e5d5d48
&transaction_type=sale3d
&terminal_token=394f2ebc3646d3c017fa1e1cbc4a1e20
&status=pending_async
&amount=500
&signature=088e16a1019277b15d58faf0541e11910eb756f6
&eci=06
&redirect_url=http://example.com/redirect_url
```

The second notification is the same as the typical 3D notifications for final statuses after the consumer has performed the 3D workflow.

## Partial Approvals

Credit cards that do not have sufficient funds in their account for the full purchase amount may be provided with a partial approval response from the issuer. When a partial approval happens, there will be a flag confirming the partial approval in the response (partial approval set to 'true') and the amount field in the response will contain the actual partially approved amount instead of the requested one in the API request. The cardholder can then choose to use a supplemental payment method to pay the balance and complete the purchase, if so desired. Merchants that accept partial approvals should note that issuers may return a partial approval response on a pre-paid/debit card at any time, and issuers may also respond with a partial approval response amount that is equal to the requested amount.

Have in mind that full/partial refunds or captures need to reference the partially approved initial transaction, you will get a workflow error if trying to capture or refund more than the partially approved amount. So make sure you check for the partial approval flag in the API response for the relevant transaction types, and handle follow-up transaction amounts properly.

With 3-D secure transactions, if a partial approval happened, the partially approved amount will be returned in the async API notification to the merchant. This is because actual communication with the acquirers happen after 3D authentication by the cardholder, in the initial API response the merchant will get the requested amount with the status 'pending async' for the transaction.

Note that partial approval support is disabled by default, feel free to contact our Risk team to enable it.

# Preauthorizations

Preauthorizations are used to request Approval for an estimated transaction amount because the final transaction amount will only be known some time later. This type of message is typically sent for transactions such as car rentals, hotel rooms and petrol. The reason for the preauthorization is to authenticate the card and the cardholder and also to check funds availability.

They are similar to the final authorizations, but have longer authorize time-frame and allow amount to be extended *(restricted per card brand)*.

**Basic Workflow**

- Preauthorization
- Incremental authorize
- Capture
- Full reversal
- Partial reversal

**PREAUTHORIZATION**

Preauthorization transaction can be submitted via normal Authorize or Authorize3d transaction with additional request param **preauthorization**.

To enable this, please contact tech-support@e-comprocessing.com.

Supported Card brands

**VISA**

Visa Lodging Preauthorization Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Authorize');
    $request = $genesis->request();

    $request
        ->setTransactionId('TrxID_24f02ae100a2a3aeca0162658321689c')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('5000')
        ->setCurrency('USD')
        ->setPreauthorization('true')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('4200000000000000')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.AuthorizeRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    AuthorizeRequest request = new AuthorizeRequest();

    request.setTransactionId("TrxID_24f02ae100a2a3aeca0162658321689c");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(5000));
    request.setCurrency("USD");
    request.setPreauthorization("true");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("4200000000000000");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.authorize(
  {
    "transaction_id": "TrxID_24f02ae100a2a3aeca0162658321689c",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": 5000,
    "currency": "USD",
    "preauthorization": true,
    "card_holder": "Travis Pastrana",
    "card_number": "4200000000000000",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>authorize</transaction_type>
    <transaction_id>TrxID_24f02ae100a2a3aeca0162658321689c</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>5000</amount>
    <currency>USD</currency>
    <preauthorization>true</preauthorization>
    <card_holder>Travis Pastrana</card_holder>
    <card_number>4200000000000000</card_number>
    <expiration_month>12</expiration_month>
    <expiration_year>2021</expiration_year>
    <cvv>834</cvv>
    <customer_email>travis@example.com</customer_email>
    <customer_phone>+1987987987987</customer_phone>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>10178</zip_code>
      <city>Los Angeles</city>
      <state>CA</state>
      <country>US</country>
    </billing_address>
  </payment_transaction>'
```

Successful Response

```
stdClass Object
(
    [transaction_type] => authorize
    [status] => approved
    [mode] => live
    [transaction_id] => TrxID_24f02ae100a2a3aeca0162658321689c
    [unique_id] => d08e0dbe0926aad0172bda9683002c29
    [avs_response_code] => 5I
    [avs_response_text] => Response provided by issuer processor; Address information not verified
    [authorization_code] => 950732
    [response_code] => 00
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:23.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 5000
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[authorize]>
    <status content=[approved]>
    <mode content=[live]>
    <transaction_id content=[TrxID_24f02ae100a2a3aeca0162658321689c]>
    <unique_id content=[d08e0dbe0926aad0172bda9683002c29]>
    <avs_response_code content=[5I]>
    <avs_response_text content=[Response provided by issuer processor; Address information not verified]>
    <authorization_code content=[950732]>
    <response_code content=[00]>
    <timestamp content=[2020-07-15T12:08:23Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[5000]>
    <currency content=[USD]>
    <sent_to_acquirer content=[true]>
]>
```

```
{
    transaction_type: "authorize",
    status: "approved",
    mode: "live",
    transaction_id: "TrxID_24f02ae100a2a3aeca0162658321689c",
    unique_id: "d08e0dbe0926aad0172bda9683002c29",
    avs_response_code: "5I",
    avs_response_text: "Response provided by issuer processor; Address information not verified",
    authorization_code: "950732",
    response_code: "00",
    timestamp: "2020-07-15T12:08:23Z",
    descriptor: "Descriptor one",
    amount: "5000",
    currency: "USD",
    sent_to_acquirer: "true",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>authorize</transaction_type>
  <status>approved</status>
  <mode>live</mode>
  <transaction_id>TrxID_24f02ae100a2a3aeca0162658321689c</transaction_id>
  <unique_id>d08e0dbe0926aad0172bda9683002c29</unique_id>
  <avs_response_code>5I</avs_response_code>
  <avs_response_text>Response provided by issuer processor; Address information not verified</avs_response_text>
  <authorization_code>950732</authorization_code>
  <response_code>00</response_code>
  <timestamp>2020-07-15T12:08:23Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>5000</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

Reconcile Visa Preauthorization Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('NonFinancial\Reconcile\Transaction');
    $request = $genesis->request();

    $request
        ->setUniqueId('d08e0dbe0926aad0172bda9683002c29');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.nonfinancial.reconcile.ReconcileRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    ReconcileRequest request = new ReconcileRequest();

    request.setUniqueId("d08e0dbe0926aad0172bda9683002c29");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.reconcile(
  {
    "unique_id": "d08e0dbe0926aad0172bda9683002c29"
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/reconcile/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <reconcile>
    <unique_id>d08e0dbe0926aad0172bda9683002c29</unique_id>
  </reconcile>'
```

Successful Preauthorization Reconciliation Response

```
stdClass Object
(
    [transaction_type] => authorize
    [status] => approved
    [authorization_code] => 950732
    [retrieval_reference_number] => 016813015184
    [response_code] => 00
    [unique_id] => d08e0dbe0926aad0172bda9683002c29
    [transaction_id] => TrxID_24f02ae100a2a3aeca0162658321689c
    [mode] => live
    [timestamp] => 2020-07-15T12:08:23Z
    [descriptor] => Descriptor one
    [amount] => 5000
    [currency] => USD
    [card_brand] => visa
    [card_number] => 420000...0000
    [card_type] => CREDIT
    [card_subtype] => CARD SUBTYPE
    [card_issuing_bank] => Issuing Bank
    [bank_account_number] => Bank Account Number
    [bank_identifier_code] => Bank Identifier Code
    [sent_to_acquirer] => true
    [arn] => 74537605259536043849425
    [preauthorization] => true
    [preauthorization_expires_at] => 2020-08-15T12:08:23Z
    [preauthorization_total_amount] => 5000
    [capturable_amount] => 5750
    [captured_amount] => 0
    [reversed_amount] => 0
    [reversible_amount] => 5000
)
```

```xml
<payment_response content=[
    <transaction_type content=[authorize]>
    <status content=[approved]>
    <authorization_code content=[950732]>
    <retrieval_reference_number content=[016813015184]>
    <response_code content=[00]>
    <unique_id content=[d08e0dbe0926aad0172bda9683002c29]>
    <transaction_id content=[TrxID_24f02ae100a2a3aeca0162658321689c]>
    <mode content=[live]>
    <timestamp content=[2020-07-15T12:08:23Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[5000]>
    <currency content=[USD]>
    <card_brand content=[visa]>
    <card_number content=[420000...0000]>
    <card_type content=[CREDIT]>
    <card_subtype content=[CARD SUBTYPE]>
    <card_issuing_bank content=[Issuing Bank]>
    <bank_account_number content=[Bank Account Number]>
    <bank_identifier_code content=[Bank Identifier Code]>
    <sent_to_acquirer content=[true]>
    <arn content=[74537605259536043849425]>
    <preauthorization content=[true]>
    <preauthorization_expires_at content=[2020-08-15T12:08:23Z]>
    <preauthorization_total_amount content=[5000]>
    <capturable_amount content=[5750]>
    <captured_amount content=[0]>
    <reversed_amount content=[0]>
    <reversible_amount content=[5000]>
]>
```

```json
{
    transaction_type: "authorize",
    status: "approved",
    authorization_code: "950732",
    retrieval_reference_number: "016813015184",
    response_code: "00",
    unique_id: "d08e0dbe0926aad0172bda9683002c29",
    transaction_id: "TrxID_24f02ae100a2a3aeca0162658321689c",
    mode: "live",
    timestamp: "2020-07-15T12:08:23Z",
    descriptor: "Descriptor one",
    amount: "5000",
    currency: "USD",
    card_brand: "visa",
    card_number: "420000...0000",
    card_type: "CREDIT",
    card_subtype: "CARD SUBTYPE",
    card_issuing_bank: "Issuing Bank",
    bank_account_number: "Bank Account Number",
    bank_identifier_code: "Bank Identifier Code",
    sent_to_acquirer: "true",
    arn: "74537605259536043849425",
    preauthorization: "true",
    preauthorization_expires_at: "2020-08-15T12:08:23Z",
    preauthorization_total_amount: "5000",
    capturable_amount: "5750",
    captured_amount: "0",
    reversed_amount: "0",
    reversible_amount: "5000",
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_response>
  <transaction_type>authorize</transaction_type>
  <status>approved</status>
  <authorization_code>950732</authorization_code>
  <retrieval_reference_number>016813015184</retrieval_reference_number>
  <response_code>00</response_code>
  <unique_id>d08e0dbe0926aad0172bda9683002c29</unique_id>
  <transaction_id>TrxID_24f02ae100a2a3aeca0162658321689c</transaction_id>
  <mode>live</mode>
  <timestamp>2020-07-15T12:08:23Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>5000</amount>
  <currency>USD</currency>
  <card_brand>visa</card_brand>
  <card_number>420000...0000</card_number>
  <card_type>CREDIT</card_type>
  <card_subtype>CARD SUBTYPE</card_subtype>
  <card_issuing_bank>Issuing Bank</card_issuing_bank>
  <bank_account_number>Bank Account Number</bank_account_number>
  <bank_identifier_code>Bank Identifier Code</bank_identifier_code>
  <sent_to_acquirer>true</sent_to_acquirer>
  <arn>74537605259536043849425</arn>
  <preauthorization>true</preauthorization>
  <preauthorization_expires_at>2020-08-15T12:08:23Z</preauthorization_expires_at>
  <preauthorization_total_amount>5000</preauthorization_total_amount>
  <capturable_amount>5750</capturable_amount>
  <captured_amount>0</captured_amount>
  <reversed_amount>0</reversed_amount>
  <reversible_amount>5000</reversible_amount>
</payment_response>
```

- MCC Restriction - **YES**
- Authorization timeframe - **31 days** *(depends on the MCC and merchant region)*
- Authorize timeframe extension - **not supported**
- Capture tolerance - *percent* or *amount*. Navigate to the Capture section to learn more.

| MCC | Segment | Authorization timeframe | Amount tolerance |
|---|---|---|---|
| 3501-3999, 7011 | Lodging | 31 days | 15% |
| 3351-3500, 7512 | Car Rental | 31 days | 15% |
| 4411 | Steamship and Cruise Lines | 31 days | 15% |
| 7513 | Truck Rentals | 7 days | 15% |
| 7033 | Trailer Parks and Campgrounds | 7 days | 15% |
| 7519 | Motor Home and Recreational Vehicle Rentals | 7 days | 15% |
| 5552 | Electric Vehicle Charging | 7 days | 15% |
| 7523 | Parking and Garages | 7 days | 15% |
| 7394 | Equipment, Tool, Furniture and Appliance Rental | 7 days | none |
| 7999 | Recreation Services | 7 days | none |
| 7996 | Amusement Parks, Carnivals, Circuses, Fortune Tellers | 7 days | none |
| 5599 | Miscellaneous Automotive, Aircraft, and Farm Equipment Dealers | 7 days | none |
| 4457 | Boat Rentals and Leasing | 7 days | none |
| 5571 | Motorcycle Shops and Dealers | 7 days | none |
| 4111 | Local and Suburban Commuter, Passenger Transportation, including Ferries | 7 days[1] | 25 USD[3] |
| 4112 | Passenger Railways | 7 days[1] | 25 USD[3] |
| 4131 | Bus Lines | 7 days[1] | 25 USD[3] |
| 5812 | Eating Places and Restaurants | end of approval day[2] | 20% |
| 5813 | Drinking Places, Bars, Taverns, Cocktail Lounges, Nightclubs, Discotheques | end of approval day[2] | 20% |
| 4121 | Taxicabs and Limousines (Card-Absent Environment only) | end of approval day[2] | 20% |

**7 days[1]** - 7 days *(3 days for US merchant region)*

**end of approval day[2]** - end of approval day *(in the acquirer's timezone)*

**25 USD[3]** - a capture with amount up to 25 USD can be requested without a need of additional incremental authorization. Just in case the authorized amount is less than 25 USD *(15 USD for merchants in the US region)*. The respective amount will be exchanged to the transaction currency in case the currency is different than USD.

**MASTERCARD**

Master Lodging Preauthorization Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Authorize');
    $request = $genesis->request();

    $request
        ->setTransactionId('TrxID_82002ed620272e47f03af292b980d770')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('5000')
        ->setCurrency('USD')
        ->setPreauthorization('true')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('5555555555554444')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.AuthorizeRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    AuthorizeRequest request = new AuthorizeRequest();

    request.setTransactionId("TrxID_82002ed620272e47f03af292b980d770");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(5000));
    request.setCurrency("USD");
    request.setPreauthorization("true");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("5555555555554444");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.authorize(
  {
    "transaction_id": "TrxID_82002ed620272e47f03af292b980d770",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": 5000,
    "currency": "USD",
    "preauthorization": true,
    "card_holder": "Travis Pastrana",
    "card_number": "5555555555554444",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>authorize</transaction_type>
    <transaction_id>TrxID_82002ed620272e47f03af292b980d770</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>5000</amount>
    <currency>USD</currency>
    <preauthorization>true</preauthorization>
    <card_holder>Travis Pastrana</card_holder>
    <card_number>5555555555554444</card_number>
    <expiration_month>12</expiration_month>
    <expiration_year>2021</expiration_year>
    <cvv>834</cvv>
    <customer_email>travis@example.com</customer_email>
    <customer_phone>+1987987987987</customer_phone>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>10178</zip_code>
      <city>Los Angeles</city>
      <state>CA</state>
      <country>US</country>
    </billing_address>
  </payment_transaction>'
```

Successful Response

```
stdClass Object
(
    [transaction_type] => authorize
    [status] => approved
    [mode] => live
    [transaction_id] => TrxID_82002ed620272e47f03af292b980d770
    [unique_id] => b9f0cbf6bfe0778ca0d6c2d7dfca302b
    [avs_response_code] => 5I
    [avs_response_text] => Response provided by issuer processor; Address information not verified
    [authorization_code] => 451671
    [response_code] => 00
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-14 12:08:23.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 5000
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[authorize]>
    <status content=[approved]>
    <mode content=[live]>
    <transaction_id content=[TrxID_82002ed620272e47f03af292b980d770]>
    <unique_id content=[b9f0cbf6bfe0778ca0d6c2d7dfca302b]>
    <avs_response_code content=[5I]>
    <avs_response_text content=[Response provided by issuer processor; Address information not verified]>
    <authorization_code content=[451671]>
    <response_code content=[00]>
    <timestamp content=[2020-07-14T12:08:23Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[5000]>
    <currency content=[USD]>
    <sent_to_acquirer content=[true]>
]>
```

```
{
    transaction_type: "authorize",
    status: "approved",
    mode: "live",
    transaction_id: "TrxID_82002ed620272e47f03af292b980d770",
    unique_id: "b9f0cbf6bfe0778ca0d6c2d7dfca302b",
    avs_response_code: "5I",
    avs_response_text: "Response provided by issuer processor; Address information not verified",
    authorization_code: "451671",
    response_code: "00",
    timestamp: "2020-07-14T12:08:23Z",
    descriptor: "Descriptor one",
    amount: "5000",
    currency: "USD",
    sent_to_acquirer: "true",
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>authorize</transaction_type>
  <status>approved</status>
  <mode>live</mode>
  <transaction_id>TrxID_82002ed620272e47f03af292b980d770</transaction_id>
  <unique_id>b9f0cbf6bfe0778ca0d6c2d7dfca302b</unique_id>
  <avs_response_code>5I</avs_response_code>
  <avs_response_text>Response provided by issuer processor; Address information not verified</avs_response_text>
  <authorization_code>451671</authorization_code>
  <response_code>00</response_code>
  <timestamp>2020-07-14T12:08:23Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>5000</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

Reconcile Master Preauthorization Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('NonFinancial\Reconcile\Transaction');
    $request = $genesis->request();

    $request
        ->setUniqueId('b9f0cbf6bfe0778ca0d6c2d7dfca302b');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.nonfinancial.reconcile.ReconcileRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    ReconcileRequest request = new ReconcileRequest();

    request.setUniqueId("b9f0cbf6bfe0778ca0d6c2d7dfca302b");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.reconcile(
  {
    "unique_id": "b9f0cbf6bfe0778ca0d6c2d7dfca302b"
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/reconcile/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <reconcile>
    <unique_id>b9f0cbf6bfe0778ca0d6c2d7dfca302b</unique_id>
  </reconcile>'
```

Successful Preauthorization Reconciliation Response

```
stdClass Object
(
    [transaction_type] => authorize
    [status] => approved
    [authorization_code] => 451671
    [retrieval_reference_number] => 016813015184
    [response_code] => 00
    [unique_id] => b9f0cbf6bfe0778ca0d6c2d7dfca302b
    [transaction_id] => TrxID_82002ed620272e47f03af292b980d770
    [mode] => live
    [timestamp] => 2020-07-14T12:08:23Z
    [descriptor] => Descriptor one
    [amount] => 5000
    [currency] => USD
    [card_brand] => master
    [card_number] => 555555...4444
    [card_type] => CREDIT
    [card_subtype] => CARD SUBTYPE
    [card_issuing_bank] => Issuing Bank
    [bank_account_number] => Bank Account Number
    [bank_identifier_code] => Bank Identifier Code
    [sent_to_acquirer] => true
    [arn] => 74537605259536043849425
    [preauthorization] => true
    [preauthorization_expires_at] => 2020-08-13T12:08:23Z
    [preauthorization_total_amount] => 5000
    [capturable_amount] => 5000
    [captured_amount] => 0
    [reversed_amount] => 0
    [reversible_amount] => 5000
)
```

```xml
<payment_response content=[
    <transaction_type content=[authorize]>
    <status content=[approved]>
    <authorization_code content=[451671]>
    <retrieval_reference_number content=[016813015184]>
    <response_code content=[00]>
    <unique_id content=[b9f0cbf6bfe0778ca0d6c2d7dfca302b]>
    <transaction_id content=[TrxID_82002ed620272e47f03af292b980d770]>
    <mode content=[live]>
    <timestamp content=[2020-07-14T12:08:23Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[5000]>
    <currency content=[USD]>
    <card_brand content=[master]>
    <card_number content=[555555...4444]>
    <card_type content=[CREDIT]>
    <card_subtype content=[CARD SUBTYPE]>
    <card_issuing_bank content=[Issuing Bank]>
    <bank_account_number content=[Bank Account Number]>
    <bank_identifier_code content=[Bank Identifier Code]>
    <sent_to_acquirer content=[true]>
    <arn content=[74537605259536043849425]>
    <preauthorization content=[true]>
    <preauthorization_expires_at content=[2020-08-13T12:08:23Z]>
    <preauthorization_total_amount content=[5000]>
    <capturable_amount content=[5000]>
    <captured_amount content=[0]>
    <reversed_amount content=[0]>
    <reversible_amount content=[5000]>
]>
```

```json
{
    transaction_type: "authorize",
    status: "approved",
    authorization_code: "451671",
    retrieval_reference_number: "016813015184",
    response_code: "00",
    unique_id: "b9f0cbf6bfe0778ca0d6c2d7dfca302b",
    transaction_id: "TrxID_82002ed620272e47f03af292b980d770",
    mode: "live",
    timestamp: "2020-07-14T12:08:23Z",
    descriptor: "Descriptor one",
    amount: "5000",
    currency: "USD",
    card_brand: "master",
    card_number: "555555...4444",
    card_type: "CREDIT",
    card_subtype: "CARD SUBTYPE",
    card_issuing_bank: "Issuing Bank",
    bank_account_number: "Bank Account Number",
    bank_identifier_code: "Bank Identifier Code",
    sent_to_acquirer: "true",
    arn: "74537605259536043849425",
    preauthorization: "true",
    preauthorization_expires_at: "2020-08-13T12:08:23Z",
    preauthorization_total_amount: "5000",
    capturable_amount: "5000",
    captured_amount: "0",
    reversed_amount: "0",
    reversible_amount: "5000",
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_response>
  <transaction_type>authorize</transaction_type>
  <status>approved</status>
  <authorization_code>451671</authorization_code>
  <retrieval_reference_number>016813015184</retrieval_reference_number>
  <response_code>00</response_code>
  <unique_id>b9f0cbf6bfe0778ca0d6c2d7dfca302b</unique_id>
  <transaction_id>TrxID_82002ed620272e47f03af292b980d770</transaction_id>
  <mode>live</mode>
  <timestamp>2020-07-14T12:08:23Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>5000</amount>
  <currency>USD</currency>
  <card_brand>master</card_brand>
  <card_number>555555...4444</card_number>
  <card_type>CREDIT</card_type>
  <card_subtype>CARD SUBTYPE</card_subtype>
  <card_issuing_bank>Issuing Bank</card_issuing_bank>
  <bank_account_number>Bank Account Number</bank_account_number>
  <bank_identifier_code>Bank Identifier Code</bank_identifier_code>
  <sent_to_acquirer>true</sent_to_acquirer>
  <arn>74537605259536043849425</arn>
  <preauthorization>true</preauthorization>
  <preauthorization_expires_at>2020-08-13T12:08:23Z</preauthorization_expires_at>
  <preauthorization_total_amount>5000</preauthorization_total_amount>
  <capturable_amount>5000</capturable_amount>
  <captured_amount>0</captured_amount>
  <reversed_amount>0</reversed_amount>
  <reversible_amount>5000</reversible_amount>
</payment_response>
```

- MCC Restriction - **NO**
- Authorize timeframe - **30 days**
- Authorize timeframe extension - **supported via Incremental authorize**
- Capture tolerance - **YES**, but only for the MCCs below. Navigate to theCapture section to learn more.

| MCC | Segment | Authorization timeframe | Amount tolerance |
|---|---|---|---|
| 5812 | Eating Places, Restaurants | 30 days | 20% |
| 5814 | Fast Food Restaurants | 30 days | 20% |

**MAESTRO**

Intl Maestro Lodging Preauthorization Request

```
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Authorize');
    $request = $genesis->request();

    $request
        ->setTransactionId('TrxID_e9f9327dfbf94c952de40be14f9a6473')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('5000')
        ->setCurrency('USD')
        ->setPreauthorization('true')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('6759411100000008')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.AuthorizeRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    AuthorizeRequest request = new AuthorizeRequest();

    request.setTransactionId("TrxID_e9f9327dfbf94c952de40be14f9a6473");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(5000));
    request.setCurrency("USD");
    request.setPreauthorization("true");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("6759411100000008");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.authorize(
  {
    "transaction_id": "TrxID_e9f9327dfbf94c952de40be14f9a6473",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": 5000,
    "currency": "USD",
    "preauthorization": true,
    "card_holder": "Travis Pastrana",
    "card_number": "6759411100000008",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>authorize</transaction_type>
    <transaction_id>TrxID_e9f9327dfbf94c952de40be14f9a6473</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>5000</amount>
    <currency>USD</currency>
    <preauthorization>true</preauthorization>
    <card_holder>Travis Pastrana</card_holder>
    <card_number>6759411100000008</card_number>
    <expiration_month>12</expiration_month>
    <expiration_year>2021</expiration_year>
    <cvv>834</cvv>
    <customer_email>travis@example.com</customer_email>
    <customer_phone>+1987987987987</customer_phone>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>10178</zip_code>
      <city>Los Angeles</city>
      <state>CA</state>
      <country>US</country>
    </billing_address>
  </payment_transaction>'
```

Successful Response

```
stdClass Object
(
    [transaction_type] => authorize
    [status] => approved
    [mode] => live
    [transaction_id] => TrxID_e9f9327dfbf94c952de40be14f9a6473
    [unique_id] => 9c24a738d159fac1be3c266ad3623e0f
    [avs_response_code] => 5I
    [avs_response_text] => Response provided by issuer processor; Address information not verified
    [authorization_code] => 174588
    [response_code] => 00
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-13 12:08:23.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 5000
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[authorize]>
    <status content=[approved]>
    <mode content=[live]>
    <transaction_id content=[TrxID_e9f9327dfbf94c952de40be14f9a6473]>
    <unique_id content=[9c24a738d159fac1be3c266ad3623e0f]>
    <avs_response_code content=[5I]>
    <avs_response_text content=[Response provided by issuer processor; Address information not verified]>
    <authorization_code content=[174588]>
    <response_code content=[00]>
    <timestamp content=[2020-07-13T12:08:23Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[5000]>
    <currency content=[USD]>
    <sent_to_acquirer content=[true]>
]>
```

```
{
    transaction_type: "authorize",
    status: "approved",
    mode: "live",
    transaction_id: "TrxID_e9f9327dfbf94c952de40be14f9a6473",
    unique_id: "9c24a738d159fac1be3c266ad3623e0f",
    avs_response_code: "5I",
    avs_response_text: "Response provided by issuer processor; Address information not verified",
    authorization_code: "174588",
    response_code: "00",
    timestamp: "2020-07-13T12:08:23Z",
    descriptor: "Descriptor one",
    amount: "5000",
    currency: "USD",
    sent_to_acquirer: "true",
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>authorize</transaction_type>
  <status>approved</status>
  <mode>live</mode>
  <transaction_id>TrxID_e9f9327dfbf94c952de40be14f9a6473</transaction_id>
  <unique_id>9c24a738d159fac1be3c266ad3623e0f</unique_id>
  <avs_response_code>5I</avs_response_code>
  <avs_response_text>Response provided by issuer processor; Address information not verified</avs_response_text>
  <authorization_code>174588</authorization_code>
  <response_code>00</response_code>
  <timestamp>2020-07-13T12:08:23Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>5000</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```

Reconcile Intl Maestro Preauthorization Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('NonFinancial\Reconcile\Transaction');
    $request = $genesis->request();

    $request
        ->setUniqueId('9c24a738d159fac1be3c266ad3623e0f');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.nonfinancial.reconcile.ReconcileRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    ReconcileRequest request = new ReconcileRequest();

    request.setUniqueId("9c24a738d159fac1be3c266ad3623e0f");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.reconcile(
  {
    "unique_id": "9c24a738d159fac1be3c266ad3623e0f"
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/reconcile/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <reconcile>
    <unique_id>9c24a738d159fac1be3c266ad3623e0f</unique_id>
  </reconcile>'
```

Successful Preauthorization Reconciliation Response

```
stdClass Object
(
    [transaction_type] => authorize
    [status] => approved
    [authorization_code] => 174588
    [retrieval_reference_number] => 016813015184
    [response_code] => 00
    [unique_id] => 9c24a738d159fac1be3c266ad3623e0f
    [transaction_id] => TrxID_e9f9327dfbf94c952de40be14f9a6473
    [mode] => live
    [timestamp] => 2020-07-13T12:08:23Z
    [descriptor] => Descriptor one
    [amount] => 5000
    [currency] => USD
    [card_brand] => Intl Maestro
    [card_number] => 675941...0008
    [card_type] => CREDIT
    [card_subtype] => CARD SUBTYPE
    [card_issuing_bank] => Issuing Bank
    [bank_account_number] => Bank Account Number
    [bank_identifier_code] => Bank Identifier Code
    [sent_to_acquirer] => true
    [arn] => 74537605259536043849425
    [preauthorization] => true
    [preauthorization_expires_at] => 2020-07-20T12:08:23Z
    [preauthorization_total_amount] => 5000
    [capturable_amount] => 5000
    [captured_amount] => 0
    [reversed_amount] => 0
    [reversible_amount] => 5000
)
```

```xml
<payment_response content=[
    <transaction_type content=[authorize]>
    <status content=[approved]>
    <authorization_code content=[174588]>
    <retrieval_reference_number content=[016813015184]>
    <response_code content=[00]>
    <unique_id content=[9c24a738d159fac1be3c266ad3623e0f]>
    <transaction_id content=[TrxID_e9f9327dfbf94c952de40be14f9a6473]>
    <mode content=[live]>
    <timestamp content=[2020-07-13T12:08:23Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[5000]>
    <currency content=[USD]>
    <card_brand content=[Intl Maestro]>
    <card_number content=[675941...0008]>
    <card_type content=[CREDIT]>
    <card_subtype content=[CARD SUBTYPE]>
    <card_issuing_bank content=[Issuing Bank]>
    <bank_account_number content=[Bank Account Number]>
    <bank_identifier_code content=[Bank Identifier Code]>
    <sent_to_acquirer content=[true]>
    <arn content=[74537605259536043849425]>
    <preauthorization content=[true]>
    <preauthorization_expires_at content=[2020-07-20T12:08:23Z]>
    <preauthorization_total_amount content=[5000]>
    <capturable_amount content=[5000]>
    <captured_amount content=[0]>
    <reversed_amount content=[0]>
    <reversible_amount content=[5000]>
]>
```

```json
{
    transaction_type: "authorize",
    status: "approved",
    authorization_code: "174588",
    retrieval_reference_number: "016813015184",
    response_code: "00",
    unique_id: "9c24a738d159fac1be3c266ad3623e0f",
    transaction_id: "TrxID_e9f9327dfbf94c952de40be14f9a6473",
    mode: "live",
    timestamp: "2020-07-13T12:08:23Z",
    descriptor: "Descriptor one",
    amount: "5000",
    currency: "USD",
    card_brand: "Intl Maestro",
    card_number: "675941...0008",
    card_type: "CREDIT",
    card_subtype: "CARD SUBTYPE",
    card_issuing_bank: "Issuing Bank",
    bank_account_number: "Bank Account Number",
    bank_identifier_code: "Bank Identifier Code",
    sent_to_acquirer: "true",
    arn: "74537605259536043849425",
    preauthorization: "true",
    preauthorization_expires_at: "2020-07-20T12:08:23Z",
    preauthorization_total_amount: "5000",
    capturable_amount: "5000",
    captured_amount: "0",
    reversed_amount: "0",
    reversible_amount: "5000",
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_response>
  <transaction_type>authorize</transaction_type>
  <status>approved</status>
  <authorization_code>174588</authorization_code>
  <retrieval_reference_number>016813015184</retrieval_reference_number>
  <response_code>00</response_code>
  <unique_id>9c24a738d159fac1be3c266ad3623e0f</unique_id>
  <transaction_id>TrxID_e9f9327dfbf94c952de40be14f9a6473</transaction_id>
  <mode>live</mode>
  <timestamp>2020-07-13T12:08:23Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>5000</amount>
  <currency>USD</currency>
  <card_brand>Intl Maestro</card_brand>
  <card_number>675941...0008</card_number>
  <card_type>CREDIT</card_type>
  <card_subtype>CARD SUBTYPE</card_subtype>
  <card_issuing_bank>Issuing Bank</card_issuing_bank>
  <bank_account_number>Bank Account Number</bank_account_number>
  <bank_identifier_code>Bank Identifier Code</bank_identifier_code>
  <sent_to_acquirer>true</sent_to_acquirer>
  <arn>74537605259536043849425</arn>
  <preauthorization>true</preauthorization>
  <preauthorization_expires_at>2020-07-20T12:08:23Z</preauthorization_expires_at>
  <preauthorization_total_amount>5000</preauthorization_total_amount>
  <capturable_amount>5000</capturable_amount>
  <captured_amount>0</captured_amount>
  <reversed_amount>0</reversed_amount>
  <reversible_amount>5000</reversible_amount>
</payment_response>
```

- MCC Restriction - **NO**
- Authorize timeframe - **7 days**
- Authorize timeframe extension - **supported via Incremental authorize**
- Capture tolerance - **YES**, but only for the MCCs below. Navigate to theCapture section to learn more.

| MCC | Segment | Authorization timeframe | Amount tolerance |
|---|---|---|---|
| 5812 | Eating Places, Restaurants | 7 days | 20% |
| 5814 | Fast Food Restaurants | 7 days | 20% |

---

Reconcile the preauthorization to retrieve the Preauthorization specifics:

- Preauthorization expiration
- Total preauthorized amount
- Capturable amount
- Captured amount

#### INCREMENTAL AUTHORIZE

Incremental authorizations are used in preauthorization workflow to:

- extend the preauthorization amount
- extend the preauthorization time-frame

> Incremental authorizations are non-3DS, because they only refer to the Preauthorization transaction. They cannot be voided / refunded etc, can only modify/extend the related preauthorization.

An incremental authorization transaction can be submitted in case:

- Preauthorization is approved and preauthorization time-frame is not expired
- Preauthorization has not been captured

> A Reconciliation could be performed to find out when a particular preauthorization is about to expire

Extend Preauthorization Timeframe & Amount

Mastercard Incremental Authorization Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Preauthorization\IncrementalAuthorize');
    $request = $genesis->request();

    $request
        ->setTransactionId('TrxID_29d074f682c3c9aa97bbca7934e30781')
        ->setUsage('20469237 extend hotel rezervation')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('1000')
        ->setReferenceId('b9f0cbf6bfe0778ca0d6c2d7dfca302b');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>incremental_authorize</transaction_type>
   <transaction_id>TrxID_29d074f682c3c9aa97bbca7934e30781</transaction_id>
   <usage>20469237 extend hotel rezervation</usage>
   <remote_ip>245.253.2.12</remote_ip>
   <amount>1000</amount>
   <reference_id>b9f0cbf6bfe0778ca0d6c2d7dfca302b</reference_id>
 </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **incremental_authorize** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | required* | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| amount | required | integer >= 0 | Topup amount in minor currency unit, see Currency and Amount Handling for details |
| reference_id | required | string(32) | Unique id of the corresponding preauthorization transaction |

`required* = conditionally required`

Incremental authorize with **zero** amount is allowed only for Mastercard and Maestro transactions. It will extend only the preauthorization timeframe, but not the preauthorized amount.

Successful Response

```
stdClass Object
(
    [transaction_type] => incremental_authorize
    [status] => approved
    [mode] => live
    [transaction_id] => TrxID_29d074f682c3c9aa97bbca7934e30781
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [avs_response_code] => 5I
    [avs_response_text] => Response provided by issuer processor; Address information not verified
    [authorization_code] => 485335
    [retrieval_reference_number] => 016813015184
    [response_code] => 00
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:24.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 1000
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>incremental_authorize</transaction_type>
   <status>approved</status>
   <mode>live</mode>
   <transaction_id>TrxID_29d074f682c3c9aa97bbca7934e30781</transaction_id>
   <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
   <avs_response_code>5I</avs_response_code>
   <avs_response_text>Response provided by issuer processor; Address information not verified</avs_response_text>
   <authorization_code>485335</authorization_code>
   <retrieval_reference_number>016813015184</retrieval_reference_number>
   <response_code>00</response_code>
   <timestamp>2020-07-15T12:08:24Z</timestamp>
   <descriptor>Descriptor one</descriptor>
   <amount>1000</amount>
   <currency>USD</currency>
   <sent_to_acquirer>true</sent_to_acquirer>
 </payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| gaming | string(255) | Signifies whether a gaming transaction is performed. Gaming transactions usually use MCC 7995, contact tech support for more details. |
| moto | string(255) | Signifies whether a MOTO (mail order telephone order) transaction is performed. Contact tech support for more details. |
| avs_response_code | string(255) | Generated by the card network on trying to match the billing address when performing the address verification. Optional, returned if config is enabled and acquirer supports it. Check AVS Status Codes for details. |
| avs_response_text | string(255) | Gives the human response text for the AVS response code above. Optional, returned if config is enabled and acquirer supports it. Check AVS Status Codes for details. |
| authorization_code | string(6) | Generated by the card network when an authorisation has occurred, used to identify that auth. Consists of 6 alphanumeric chars |
| retrieval_reference_number | string(255) | A reference number used for tracking all messages related to a given cardholder transaction returned by some acquirers. |
| response_code | string(2) | Defines the issuer result of a transaction, the status of a message or some action taken or required. See issuer response codes for details |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08- 30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| partial_approval | string(4) | Optional, set to 'true' if partial approval happened. Partially approved amount is then in the amount field. Check Partial Approvals for details |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [transaction_type] => incremental_authorize
    [status] => error
    [mode] => live
    [transaction_id] => TrxID_29d074f682c3c9aa97bbca7934e30781
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 400
    [technical_message] => Preauthorization has been captured, no further incremental authorizations allowed
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:24.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 1000
    [currency] => USD
    [sent_to_acquirer] => false
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>incremental_authorize</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>TrxID_29d074f682c3c9aa97bbca7934e30781</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>400</code>
  <technical_message>Preauthorization has been captured, no further incremental authorizations allowed</technical_message>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:24Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>1000</amount>
  <currency>USD</currency>
  <sent_to_acquirer>false</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| gaming | string(255) | Signifies whether a gaming transaction is performed. Gaming transactions usually use MCC 7995, contact tech support for more details. |
| moto | string(255) | Signifies whether a MOTO (mail order telephone order) transaction is performed. Contact tech support for more details. |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08- 30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| dynamic_descriptor_params | section | Optional, returned only if dynamic desc params are submitted on the API. Note here that the formatted dyn desc params are returned - as they would be submitted to the schemes for settlement. |

Example Xml For Extending The Time Frame Only

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Preauthorization\IncrementalAuthorize');
    $request = $genesis->request();

    $request
        ->setTransactionId('TrxID_29d074f682c3c9aa97bbca7934e30781')
        ->setUsage('204692378 extend the preauthorization validity timeframe')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('0')
        ->setReferenceId('d08e0dbe0926aad0172bda9683002c29');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```curl
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>incremental_authorize</transaction_type>
    <transaction_id>TrxID_29d074f682c3c9aa97bbca7934e30781</transaction_id>
    <usage>204692378 extend the preauthorization validity timeframe</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>0</amount>
    <reference_id>d08e0dbe0926aad0172bda9683002c29</reference_id>
  </payment_transaction>'
```

Error Response

```
stdClass Object
(
    [transaction_type] => incremental_authorize
    [status] => error
    [mode] => live
    [transaction_id] => TrxID_29d074f682c3c9aa97bbca7934e30781
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 400
    [technical_message] => Incremental authorizations with no financial impact are currently not supported by card brand
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:24.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 0
    [currency] => USD
    [sent_to_acquirer] => false
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>incremental_authorize</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>TrxID_29d074f682c3c9aa97bbca7934e30781</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>400</code>
  <technical_message>Incremental authorizations with no financial impact are currently not supported by card brand</technical_message>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:24Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>0</amount>
  <currency>USD</currency>
  <sent_to_acquirer>false</sent_to_acquirer>
</payment_transaction>
```

Reconcile Preauthorization By Unique Id Request

```
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('NonFinancial\Reconcile\Transaction');
    $request = $genesis->request();

    $request
        ->setUniqueId('b9f0cbf6bfe0778ca0d6c2d7dfca302b');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.nonfinancial.reconcile.ReconcileRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    ReconcileRequest request = new ReconcileRequest();

    request.setUniqueId("b9f0cbf6bfe0778ca0d6c2d7dfca302b");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.reconcile(
  {
    "unique_id": "b9f0cbf6bfe0778ca0d6c2d7dfca302b"
  }
).send()
  .then(success)
  .catch(failure);
```

```bash
curl https://username:password@staging.gate.e-comprocessing.net/reconcile/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <reconcile>
    <unique_id>b9f0cbf6bfe0778ca0d6c2d7dfca302b</unique_id>
  </reconcile>'
```

Preauthorization time-frame will be extended for Mastercard & Maestro transactions, but not for VISA

Successful Preauthorization Reconciliation Response

```
stdClass Object
(
    [transaction_type] => authorize
    [status] => approved
    [authorization_code] => 485335
    [retrieval_reference_number] => 016813015184
    [response_code] => 00
    [unique_id] => b9f0cbf6bfe0778ca0d6c2d7dfca302b
    [transaction_id] => TrxID_29d074f682c3c9aa97bbca7934e30781
    [mode] => live
    [timestamp] => 2020-07-15T12:08:24Z
    [descriptor] => Descriptor one
    [amount] => 5000
    [currency] => USD
    [card_brand] => master
    [card_number] => 555555...4444
    [card_type] => CREDIT
    [card_subtype] => CARD SUBTYPE
    [card_issuing_bank] => Issuing Bank
    [bank_account_number] => Bank Account Number
    [bank_identifier_code] => Bank Identifier Code
    [sent_to_acquirer] => true
    [arn] => 74537605259536043849425
    [preauthorization] => true
    [preauthorization_expires_at] => 2020-09-12T12:08:23Z
    [preauthorization_total_amount] => 6000
    [capturable_amount] => 6000
    [captured_amount] => 0
)
```

```
<payment_response content=[
    <transaction_type content=[authorize]>
    <status content=[approved]>
    <authorization_code content=[485335]>
    <retrieval_reference_number content=[016813015184]>
    <response_code content=[00]>
    <unique_id content=[b9f0cbf6bfe0778ca0d6c2d7dfca302b]>
    <transaction_id content=[TrxID_29d074f682c3c9aa97bbca7934e30781]>
    <mode content=[live]>
    <timestamp content=[2020-07-15T12:08:24Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[5000]>
    <currency content=[USD]>
    <card_brand content=[master]>
    <card_number content=[555555...4444]>
    <card_type content=[CREDIT]>
    <card_subtype content=[CARD SUBTYPE]>
    <card_issuing_bank content=[Issuing Bank]>
    <bank_account_number content=[Bank Account Number]>
    <bank_identifier_code content=[Bank Identifier Code]>
    <sent_to_acquirer content=[true]>
    <arn content=[74537605259536043849425]>
    <preauthorization content=[true]>
    <preauthorization_expires_at content=[2020-09-12T12:08:23Z]>
    <preauthorization_total_amount content=[6000]>
    <capturable_amount content=[6000]>
    <captured_amount content=[0]>
]>
```

```
{
    transaction_type: "authorize",
    status: "approved",
    authorization_code: "485335",
    retrieval_reference_number: "016813015184",
    response_code: "00",
    unique_id: "b9f0cbf6bfe0778ca0d6c2d7dfca302b",
    transaction_id: "TrxID_29d074f682c3c9aa97bbca7934e30781",
    mode: "live",
    timestamp: "2020-07-15T12:08:24Z",
    descriptor: "Descriptor one",
    amount: "5000",
    currency: "USD",
    card_brand: "master",
    card_number: "555555...4444",
    card_type: "CREDIT",
    card_subtype: "CARD SUBTYPE",
    card_issuing_bank: "Issuing Bank",
    bank_account_number: "Bank Account Number",
    bank_identifier_code: "Bank Identifier Code",
    sent_to_acquirer: "true",
    arn: "74537605259536043849425",
    preauthorization: "true",
    preauthorization_expires_at: "2020-09-12T12:08:23Z",
    preauthorization_total_amount: "6000",
    capturable_amount: "6000",
    captured_amount: "0",
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_response>
  <transaction_type>authorize</transaction_type>
  <status>approved</status>
  <authorization_code>485335</authorization_code>
  <retrieval_reference_number>016813015184</retrieval_reference_number>
  <response_code>00</response_code>
  <unique_id>b9f0cbf6bfe0778ca0d6c2d7dfca302b</unique_id>
  <transaction_id>TrxID_29d074f682c3c9aa97bbca7934e30781</transaction_id>
  <mode>live</mode>
  <timestamp>2020-07-15T12:08:24Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>5000</amount>
  <currency>USD</currency>
  <card_brand>master</card_brand>
  <card_number>555555...4444</card_number>
  <card_type>CREDIT</card_type>
  <card_subtype>CARD SUBTYPE</card_subtype>
  <card_issuing_bank>Issuing Bank</card_issuing_bank>
  <bank_account_number>Bank Account Number</bank_account_number>
  <bank_identifier_code>Bank Identifier Code</bank_identifier_code>
  <sent_to_acquirer>true</sent_to_acquirer>
  <arn>74537605259536043849425</arn>
  <preauthorization>true</preauthorization>
  <preauthorization_expires_at>2020-09-12T12:08:23Z</preauthorization_expires_at>
  <preauthorization_total_amount>6000</preauthorization_total_amount>
  <capturable_amount>6000</capturable_amount>
  <captured_amount>0</captured_amount>
</payment_response>
```

**CAPTURE**

Preauthorizations can be captured using the standard Capture transaction with the correct `reference_id` of the Preauthorization.

> A preauthorization can be captured only once. It can be a full & partial capture, but not multiple partial capture. In case of partial capture, the amount left needs to be reversed / returned first to the consumer using Partial Reversal transaction.

> The capturable amount can be greater than the total authorized amount (only for VISA), check Preauthorization section for more info. The amount tolerance is supported only for VISA transactions, can be defined per percent or amount.

An additional incremental authorization will be necessary when the requested capture amount is greater than:

- the total authorized amount including the calculated amount tolerance (% of the total authorized amount)
- the predefined amount tolerance in value exchanged to the appropriate currency (check the amount tolerance mer MCC in the Preauthorization section)

> Multiple partial captures are not allowed for both supported card brands

Reconcile Visa Preauthorization Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('NonFinancial\Reconcile\Transaction');
    $request = $genesis->request();

    $request
        ->setUniqueId('d08e0dbe0926aad0172bda9683002c29');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.nonfinancial.reconcile.ReconcileRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    ReconcileRequest request = new ReconcileRequest();

    request.setUniqueId("d08e0dbe0926aad0172bda9683002c29");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.reconcile(
  {
    "unique_id": "d08e0dbe0926aad0172bda9683002c29"
  }
).send()
  .then(success)
  .catch(failure);
```

```bash
curl https://username:password@staging.gate.e-comprocessing.net/reconcile/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <reconcile>
    <unique_id>d08e0dbe0926aad0172bda9683002c29</unique_id>
  </reconcile>'
```

Successful Preauthorization Reconciliation Response

```
stdClass Object
(
    [transaction_type] => authorize
    [status] => approved
    [authorization_code] => 005645
    [retrieval_reference_number] => 016813015184
    [response_code] => 00
    [unique_id] => d08e0dbe0926aad0172bda9683002c29
    [transaction_id] => 119643250547501c79d8295
    [mode] => live
    [timestamp] => 2020-07-15T12:08:23Z
    [descriptor] => Descriptor one
    [amount] => 3000
    [currency] => USD
    [card_brand] => visa
    [card_number] => 420000...0000
    [card_type] => CREDIT
    [card_subtype] => CARD SUBTYPE
    [card_issuing_bank] => Issuing Bank
    [bank_account_number] => Bank Account Number
    [bank_identifier_code] => Bank Identifier Code
    [sent_to_acquirer] => true
    [arn] => 74537605259536043849425
    [preauthorization] => true
    [preauthorization_expires_at] => 2020-08-15T12:08:23Z
    [preauthorization_total_amount] => 5000
    [capturable_amount] => 5750
    [captured_amount] => 0
    [reversed_amount] => 0
    [reversible_amount] => 5000
)
```

```
<payment_response content=[
    <transaction_type content=[authorize]>
    <status content=[approved]>
    <authorization_code content=[005645]>
    <retrieval_reference_number content=[016813015184]>
    <response_code content=[00]>
    <unique_id content=[d08e0dbe0926aad0172bda9683002c29]>
    <transaction_id content=[119643250547501c79d8295]>
    <mode content=[live]>
    <timestamp content=[2020-07-15T12:08:23Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[3000]>
    <currency content=[USD]>
    <card_brand content=[visa]>
    <card_number content=[420000...0000]>
    <card_type content=[CREDIT]>
    <card_subtype content=[CARD SUBTYPE]>
    <card_issuing_bank content=[Issuing Bank]>
    <bank_account_number content=[Bank Account Number]>
    <bank_identifier_code content=[Bank Identifier Code]>
    <sent_to_acquirer content=[true]>
    <arn content=[74537605259536043849425]>
    <preauthorization content=[true]>
    <preauthorization_expires_at content=[2020-08-15T12:08:23Z]>
    <preauthorization_total_amount content=[5000]>
    <capturable_amount content=[5750]>
    <captured_amount content=[0]>
    <reversed_amount content=[0]>
    <reversible_amount content=[5000]>
]>
```

```
{
    transaction_type: "authorize",
    status: "approved",
    authorization_code: "005645",
    retrieval_reference_number: "016813015184",
    response_code: "00",
    unique_id: "d08e0dbe0926aad0172bda9683002c29",
    transaction_id: "119643250547501c79d8295",
    mode: "live",
    timestamp: "2020-07-15T12:08:23Z",
    descriptor: "Descriptor one",
    amount: "3000",
    currency: "USD",
    card_brand: "visa",
    card_number: "420000...0000",
    card_type: "CREDIT",
    card_subtype: "CARD SUBTYPE",
    card_issuing_bank: "Issuing Bank",
    bank_account_number: "Bank Account Number",
    bank_identifier_code: "Bank Identifier Code",
    sent_to_acquirer: "true",
    arn: "74537605259536043849425",
    preauthorization: "true",
    preauthorization_expires_at: "2020-08-15T12:08:23Z",
    preauthorization_total_amount: "5000",
    capturable_amount: "5750",
    captured_amount: "0",
    reversed_amount: "0",
    reversible_amount: "5000",
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_response>
  <transaction_type>authorize</transaction_type>
  <status>approved</status>
  <authorization_code>005645</authorization_code>
  <retrieval_reference_number>016813015184</retrieval_reference_number>
  <response_code>00</response_code>
  <unique_id>d08e0dbe0926aad0172bda9683002c29</unique_id>
  <transaction_id>119643250547501c79d8295</transaction_id>
  <mode>live</mode>
  <timestamp>2020-07-15T12:08:23Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>3000</amount>
  <currency>USD</currency>
  <card_brand>visa</card_brand>
  <card_number>420000...0000</card_number>
  <card_type>CREDIT</card_type>
  <card_subtype>CARD SUBTYPE</card_subtype>
  <card_issuing_bank>Issuing Bank</card_issuing_bank>
  <bank_account_number>Bank Account Number</bank_account_number>
  <bank_identifier_code>Bank Identifier Code</bank_identifier_code>
  <sent_to_acquirer>true</sent_to_acquirer>
  <arn>74537605259536043849425</arn>
  <preauthorization>true</preauthorization>
  <preauthorization_expires_at>2020-08-15T12:08:23Z</preauthorization_expires_at>
  <preauthorization_total_amount>5000</preauthorization_total_amount>
  <capturable_amount>5750</capturable_amount>
  <captured_amount>0</captured_amount>
  <reversed_amount>0</reversed_amount>
  <reversible_amount>5000</reversible_amount>
</payment_response>
```


Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Capture');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('Check out from the Hotel')
        ->setRemoteIp('245.253.2.12')
        ->setReferenceId('43672')
        ->setAmount('5500')
        ->setCurrency('USD');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.CaptureRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    CaptureRequest request = new CaptureRequest();

    request.setTransactionId("43671");
    request.setUsage("Check out from the Hotel");
    request.setRemoteIp("245.253.2.12");
    request.setReferenceId("43672");
    request.setAmount(new BigDecimal(5500));
    request.setCurrency("USD");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.capture(
  {
    "transaction_id": "43671",
    "usage": "Check out from the Hotel",
    "remote_ip": "245.253.2.12",
    "reference_id": "43672",
    "amount": 5500,
    "currency": "USD"
  }
).send()
  .then(success)
  .catch(failure);
```

```curl
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>capture</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>Check out from the Hotel</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <reference_id>43672</reference_id>
    <amount>5500</amount>
    <currency>USD</currency>
  </payment_transaction>'
```

Successful Response

```
stdClass Object
(
    [transaction_type] => capture
    [status] => approved
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [authorization_code] => 345678
    [response_code] => 00
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:24.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 5500
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
<payment_response content=[
    <transaction_type content=[capture]>
    <status content=[approved]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <authorization_code content=[345678]>
    <response_code content=[00]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <timestamp content=[2020-07-15T12:08:24Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[5500]>
    <currency content=[USD]>
    <sent_to_acquirer content=[true]>
]>
```

```
{
    transaction_type: "capture",
    status: "approved",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    authorization_code: "345678",
    response_code: "00",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    timestamp: "2020-07-15T12:08:24Z",
    descriptor: "Descriptor one",
    amount: "5500",
    currency: "USD",
    sent_to_acquirer: "true",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>capture</transaction_type>
  <status>approved</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <authorization_code>345678</authorization_code>
  <response_code>00</response_code>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <timestamp>2020-07-15T12:08:24Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>5500</amount>
  <currency>USD</currency>
  <sent_to_acquirer>true</sent_to_acquirer>
</payment_transaction>
```


Error Response

```
stdClass Object
(
    [transaction_type] => capture
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 420
    [technical_message] => partial reversal is required first for the rest amount
    [message] => Transaction declined.
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:24.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 4000
    [currency] => USD
    [sent_to_acquirer] => false
)
```

```
<payment_response content=[
    <transaction_type content=[capture]>
    <status content=[error]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <code content=[420]>
    <technical_message content=[partial reversal is required first for the rest amount]>
    <message content=[Transaction declined.]>
    <timestamp content=[2020-07-15T12:08:24Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[4000]>
    <currency content=[USD]>
    <sent_to_acquirer content=[false]>
]>
```

```
{
    transaction_type: "capture",
    status: "error",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    code: "420",
    technical_message: "partial reversal is required first for the rest amount",
    message: "Transaction declined.",
    timestamp: "2020-07-15T12:08:24Z",
    descriptor: "Descriptor one",
    amount: "4000",
    currency: "USD",
    sent_to_acquirer: "false",
}
```

```
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>capture</transaction_type>
    <status>error</status>
    <mode>live</mode>
    <transaction_id>43671</transaction_id>
    <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
    <code>420</code>
    <technical_message>partial reversal is required first for the rest amount</technical_message>
    <message>Transaction declined.</message>
    <timestamp>2020-07-15T12:08:24Z</timestamp>
    <descriptor>Descriptor one</descriptor>
    <amount>4000</amount>
    <currency>USD</currency>
    <sent_to_acquirer>false</sent_to_acquirer>
  </payment_transaction>
```

Reconcile Visa Preauthorization Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('NonFinancial\Reconcile\Transaction');
    $request = $genesis->request();

    $request
        ->setUniqueId('d08e0dbe0926aad0172bda9683002c29');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.nonfinancial.reconcile.ReconcileRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    ReconcileRequest request = new ReconcileRequest();

    request.setUniqueId("d08e0dbe0926aad0172bda9683002c29");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```node
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.reconcile(
  {
    "unique_id": "d08e0dbe0926aad0172bda9683002c29"
  }
).send()
  .then(success)
  .catch(failure);
```

```bash
curl https://username:password@staging.gate.e-comprocessing.net/reconcile/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <reconcile>
    <unique_id>d08e0dbe0926aad0172bda9683002c29</unique_id>
  </reconcile>'
```

Successful Preauthorization Reconciliation Response

```
stdClass Object
(
    [transaction_type] => authorize
    [status] => approved
    [authorization_code] => 005645
    [retrieval_reference_number] => 016813015184
    [response_code] => 00
    [unique_id] => d08e0dbe0926aad0172bda9683002c29
    [transaction_id] => 119643250547501c79d8295
    [mode] => live
    [timestamp] => 2020-07-15T12:08:23Z
    [descriptor] => Descriptor one
    [amount] => 3000
    [currency] => USD
    [card_brand] => visa
    [card_number] => 420000...0000
    [card_type] => CREDIT
    [card_subtype] => CARD SUBTYPE
    [card_issuing_bank] => Issuing Bank
    [bank_account_number] => Bank Account Number
    [bank_identifier_code] => Bank Identifier Code
    [sent_to_acquirer] => true
    [arn] => 74537605259536043849425
    [preauthorization] => true
    [preauthorization_expires_at] => 2020-08-15T12:08:23Z
    [preauthorization_total_amount] => 5000
    [capturable_amount] => 0
    [captured_amount] => 5500
    [reversed_amount] => 0
    [reversible_amount] => 0
)
```

```
<payment_response content=[
    <transaction_type content=[authorize]>
    <status content=[approved]>
    <authorization_code content=[005645]>
    <retrieval_reference_number content=[016813015184]>
    <response_code content=[00]>
    <unique_id content=[d08e0dbe0926aad0172bda9683002c29]>
    <transaction_id content=[119643250547501c79d8295]>
    <mode content=[live]>
    <timestamp content=[2020-07-15T12:08:23Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[3000]>
    <currency content=[USD]>
    <card_brand content=[visa]>
    <card_number content=[420000...0000]>
    <card_type content=[CREDIT]>
    <card_subtype content=[CARD SUBTYPE]>
    <card_issuing_bank content=[Issuing Bank]>
    <bank_account_number content=[Bank Account Number]>
    <bank_identifier_code content=[Bank Identifier Code]>
    <sent_to_acquirer content=[true]>
    <arn content=[74537605259536043849425]>
    <preauthorization content=[true]>
    <preauthorization_expires_at content=[2020-08-15T12:08:23Z]>
    <preauthorization_total_amount content=[5000]>
    <capturable_amount content=[0]>
    <captured_amount content=[5500]>
    <reversed_amount content=[0]>
    <reversible_amount content=[0]>
]>
```


```
{
    transaction_type: "authorize",
    status: "approved",
    authorization_code: "005645",
    retrieval_reference_number: "016813015184",
    response_code: "00",
    unique_id: "d08e0dbe0926aad0172bda9683002c29",
    transaction_id: "119643250547501c79d8295",
    mode: "live",
    timestamp: "2020-07-15T12:08:23Z",
    descriptor: "Descriptor one",
    amount: "3000",
    currency: "USD",
    card_brand: "visa",
    card_number: "420000...0000",
    card_type: "CREDIT",
    card_subtype: "CARD SUBTYPE",
    card_issuing_bank: "Issuing Bank",
    bank_account_number: "Bank Account Number",
    bank_identifier_code: "Bank Identifier Code",
    sent_to_acquirer: "true",
    arn: "74537605259536043849425",
    preauthorization: "true",
    preauthorization_expires_at: "2020-08-15T12:08:23Z",
    preauthorization_total_amount: "5000",
    capturable_amount: "0",
    captured_amount: "5500",
    reversed_amount: "0",
    reversible_amount: "0",
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_response>
  <transaction_type>authorize</transaction_type>
  <status>approved</status>
  <authorization_code>005645</authorization_code>
  <retrieval_reference_number>016813015184</retrieval_reference_number>
  <response_code>00</response_code>
  <unique_id>d08e0dbe0926aad0172bda9683002c29</unique_id>
  <transaction_id>1196432505475 01c79d8295</transaction_id>
  <mode>live</mode>
  <timestamp>2020-07-15T12:08:23Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>3000</amount>
  <currency>USD</currency>
  <card_brand>visa</card_brand>
  <card_number>420000...0000</card_number>
  <card_type>CREDIT</card_type>
  <card_subtype>CARD SUBTYPE</card_subtype>
  <card_issuing_bank>Issuing Bank</card_issuing_bank>
  <bank_account_number>Bank Account Number</bank_account_number>
  <bank_identifier_code>Bank Identifier Code</bank_identifier_code>
  <sent_to_acquirer>true</sent_to_acquirer>
  <arn>74537605259536043849425</arn>
  <preauthorization>true</preauthorization>
  <preauthorization_expires_at>2020-08-15T12:08:23Z</preauthorization_expires_at>
  <preauthorization_total_amount>5000</preauthorization_total_amount>
  <capturable_amount>0</capturable_amount>
  <captured_amount>5500</captured_amount>
  <reversed_amount>0</reversed_amount>
  <reversible_amount>0</reversible_amount>
</payment_response>
```


**FULL REVERSAL**

Full reversal of a preauthorization can be submitted using the standardVoid transaction.

> If a preauthorization has not been captured/cleared, the merchant must ensure to submit a full reversal not later than 24 hours after the preauthorization has expired, otherwise a full reversal will be automatically performed by Genesis.

> A preauthorization can be fully reversed via the standardVoid transaction only if it has not been captured or partially reversed viaPartial Reversal transaction.

Reconcile request can be used to determine when the preauthorization is about to expire.

Reconcile Master Preauthorization Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('NonFinancial\Reconcile\Transaction');
    $request = $genesis->request();

    $request
        ->setUniqueId('b9f0cbf6bfe0778ca0d6c2d7dfca302b');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.nonfinancial.reconcile.ReconcileRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    ReconcileRequest request = new ReconcileRequest();

    request.setUniqueId("b9f0cbf6bfe0778ca0d6c2d7dfca302b");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.reconcile(
  {
    "unique_id": "b9f0cbf6bfe0778ca0d6c2d7dfca302b"
  }
).send()
  .then(success)
  .catch(failure);
```

```bash
curl https://username:password@staging.gate.e-comprocessing.net/reconcile/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <reconcile>
    <unique_id>b9f0cbf6bfe0778ca0d6c2d7dfca302b</unique_id>
  </reconcile>'
```

Successful Preauthorization Reconciliation Response

```
stdClass Object
(
    [transaction_type] => authorize
    [status] => approved
    [authorization_code] => 005645
    [retrieval_reference_number] => 016813015184
    [response_code] => 00
    [unique_id] => b9f0cbf6bfe0778ca0d6c2d7dfca302b
    [transaction_id] => 119643250547501c79d8295
    [mode] => live
    [timestamp] => 2020-07-14T12:08:23Z
    [descriptor] => Descriptor one
    [amount] => 3000
    [currency] => USD
    [card_brand] => master
    [card_number] => 555555...4444
    [card_type] => CREDIT
    [card_subtype] => CARD SUBTYPE
    [card_issuing_bank] => Issuing Bank
    [bank_account_number] => Bank Account Number
    [bank_identifier_code] => Bank Identifier Code
    [sent_to_acquirer] => true
    [arn] => 74537605259536043849425
    [preauthorization] => true
    [preauthorization_expires_at] => 2020-08-13T12:08:23Z
    [preauthorization_total_amount] => 5000
    [capturable_amount] => 5000
    [captured_amount] => 0
    [reversed_amount] => 0
    [reversible_amount] => 5000
)
```

```
<payment_response content=[
    <transaction_type content=[authorize]>
    <status content=[approved]>
    <authorization_code content=[005645]>
    <retrieval_reference_number content=[016813015184]>
    <response_code content=[00]>
    <unique_id content=[b9f0cbf6bfe0778ca0d6c2d7dfca302b]>
    <transaction_id content=[119643250547501c79d8295]>
    <mode content=[live]>
    <timestamp content=[2020-07-14T12:08:23Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[3000]>
    <currency content=[USD]>
    <card_brand content=[master]>
    <card_number content=[555555...4444]>
    <card_type content=[CREDIT]>
    <card_subtype content=[CARD SUBTYPE]>
    <card_issuing_bank content=[Issuing Bank]>
    <bank_account_number content=[Bank Account Number]>
    <bank_identifier_code content=[Bank Identifier Code]>
    <sent_to_acquirer content=[true]>
    <arn content=[74537605259536043849425]>
    <preauthorization content=[true]>
    <preauthorization_expires_at content=[2020-08-13T12:08:23Z]>
    <preauthorization_total_amount content=[5000]>
    <capturable_amount content=[5000]>
    <captured_amount content=[0]>
    <reversed_amount content=[0]>
    <reversible_amount content=[5000]>
]>
```

```
{
    transaction_type: "authorize",
    status: "approved",
    authorization_code: "005645",
    retrieval_reference_number: "016813015184",
    response_code: "00",
    unique_id: "b9f0cbf6bfe0778ca0d6c2d7dfca302b",
    transaction_id: "119643250547501c79d8295",
    mode: "live",
    timestamp: "2020-07-14T12:08:23Z",
    descriptor: "Descriptor one",
    amount: "3000",
    currency: "USD",
    card_brand: "master",
    card_number: "555555...4444",
    card_type: "CREDIT",
    card_subtype: "CARD SUBTYPE",
    card_issuing_bank: "Issuing Bank",
    bank_account_number: "Bank Account Number",
    bank_identifier_code: "Bank Identifier Code",
    sent_to_acquirer: "true",
    arn: "74537605259536043849425",
    preauthorization: "true",
    preauthorization_expires_at: "2020-08-13T12:08:23Z",
    preauthorization_total_amount: "5000",
    capturable_amount: "5000",
    captured_amount: "0",
    reversed_amount: "0",
    reversible_amount: "5000",
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_response>
  <transaction_type>authorize</transaction_type>
  <status>approved</status>
  <authorization_code>005645</authorization_code>
  <retrieval_reference_number>016813015184</retrieval_reference_number>
  <response_code>00</response_code>
  <unique_id>b9f0cbf6bfe0778ca0d6c2d7dfca302b</unique_id>
  <transaction_id>1196432505475 01c79d8295</transaction_id>
  <mode>live</mode>
  <timestamp>2020-07-14T12:08:23Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>3000</amount>
  <currency>USD</currency>
  <card_brand>master</card_brand>
  <card_number>555555...4444</card_number>
  <card_type>CREDIT</card_type>
  <card_subtype>CARD SUBTYPE</card_subtype>
  <card_issuing_bank>Issuing Bank</card_issuing_bank>
  <bank_account_number>Bank Account Number</bank_account_number>
  <bank_identifier_code>Bank Identifier Code</bank_identifier_code>
  <sent_to_acquirer>true</sent_to_acquirer>
  <arn>74537605259536043849425</arn>
  <preauthorization>true</preauthorization>
  <preauthorization_expires_at>2020-08-13T12:08:23Z</preauthorization_expires_at>
  <preauthorization_total_amount>5000</preauthorization_total_amount>
  <capturable_amount>5000</capturable_amount>
  <captured_amount>0</captured_amount>
  <reversed_amount>0</reversed_amount>
  <reversible_amount>5000</reversible_amount>
</payment_response>
```


**PARTIAL REVERSAL**

Partial reversal transactions are used in the preauthorization workflow to release a part of the total authorized amount. A transaction of this type should refer to the preauthorization directly.

> The partial reversals can be submitted no later than 24 hours after the preauthorization is about to expire.

> Reconcile could be performed to retrieve the reversible amount of a preauthorization. If a VISA Preauthorization has already been captured with a higher amount than the total preauthorized (benefiting from VISA amount tolerance), the reversible amount will be **0**. Otherwise, **reversible amount** = **preauthorization total amount** - **captured amount** - **reversed amount**.

Extend Preauthorization Timeframe & Amount

Partial Reversal Request Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Preauthorization\PartialReversal');
    $request = $genesis->request();

    $request
        ->setTransactionId('TrxID_eee0fbd400daad76b62b61471ae0d557')
        ->setUsage('40208 hotel reservation changed')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('500')
        ->setReferenceId('b9f0cbf6bfe0778ca0d6c2d7dfca302b');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>partial_reversal</transaction_type>
   <transaction_id>TrxID_eee0fbd400daad76b62b61471ae0d557</transaction_id>
   <usage>40208 hotel reservation changed</usage>
   <remote_ip>245.253.2.12</remote_ip>
   <amount>500</amount>
   <reference_id>b9f0cbf6bfe0778ca0d6c2d7dfca302b</reference_id>
 </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_type | required | string(255) | The transaction type: **partial_reversal** |
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | required* | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| remote_ip | required* | IPv4 or IPv6 address | IPv4 or IPv6 address of customer |
| amount | required | integer > 0 | The amount to be reversed in minor currency unit, seeCurrency and Amount Handling for details. |
| reference_id | required | string(32) | Unique id of the corresponding preauthorization transaction |

`required* = conditionally required`

Successful Response

```
stdClass Object
(
    [transaction_type] => partial_reversal
    [status] => approved
    [authorization_code] => 629324
    [retrieval_reference_number] => 016813015184
    [response_code] => 00
    [transaction_id] => TrxID_eee0fbd400daad76b62b61471ae0d557
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:24.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [mode] => live
    [descriptor] => Descriptor one
    [amount] => 500
    [currency] => USD
    [sent_to_acquirer] => true
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <transaction_type>partial_reversal</transaction_type>
   <status>approved</status>
   <authorization_code>629324</authorization_code>
   <retrieval_reference_number>016813015184</retrieval_reference_number>
   <response_code>00</response_code>
   <transaction_id>TrxID_eee0fbd400daad76b62b61471ae0d557</transaction_id>
   <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
   <timestamp>2020-07-15T12:08:24Z</timestamp>
   <mode>live</mode>
   <descriptor>Descriptor one</descriptor>
   <amount>500</amount>
   <currency>USD</currency>
   <sent_to_acquirer>true</sent_to_acquirer>
 </payment_transaction>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |

| Parameter | Type | Description |
|---|---|---|
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| gaming | string(255) | Signifies whether a gaming transaction is performed. Gaming transactions usually use MCC 7995, contact tech support for more details. |
| moto | string(255) | Signifies whether a MOTO (mail order telephone order) transaction is performed. Contact tech support for more details. |
| avs_response_code | string(255) | Generated by the card network on trying to match the billing address when performing the address verification. Optional, returned if config is enabled and acquirer supports it. Check AVS Status Codes for details. |
| avs_response_text | string(255) | Gives the human response text for the AVS response code above. Optional, returned if config is enabled and acquirer supports it. Check AVS Status Codes for details. |
| authorization_code | string(6) | Generated by the card network when an authorisation has occurred, used to identify that auth. Consists of 6 alphanumeric chars |
| retrieval_reference_number | string(255) | A reference number used for tracking all messages related to a given cardholder transaction returned by some acquirers. |
| response_code | string(2) | Defines the issuer result of a transaction, the status of a message or some action taken or required. See issuer response codes for details |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08- 30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| partial_approval | string(4) | Optional, set to 'true' if partial approval happened. Partially approved amount is then in the amount field. Check Partial Approvals for details |
| sent_to_acquirer | string(255) | "true" or "false" |

Error Response

```
stdClass Object
(
    [transaction_type] => partial_reversal
    [status] => error
    [mode] => live
    [transaction_id] => TrxID_eee0fbd400daad76b62b61471ae0d557
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 410
    [technical_message] => No approved preauthorization reference transaction found
    [message] => Something went wrong, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:24.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [descriptor] => Descriptor one
    [amount] => 500
    [currency] => USD
    [sent_to_acquirer] => false
)
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<payment_transaction>
  <transaction_type>partial_reversal</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>TrxID_eee0fbd400daad76b62b61471ae0d557</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>410</code>
  <technical_message>No approved preauthorization reference transaction found</technical_message>
  <message>Something went wrong, please contact support!</message>
  <timestamp>2020-07-15T12:08:24Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>500</amount>
  <currency>USD</currency>
  <sent_to_acquirer>false</sent_to_acquirer>
</payment_transaction>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |

| Parameter | Type | Description |
|---|---|---|
| status | string(255) | Status of the transaction, see states |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08-30T17:46:11Z |
| descriptor | string(255) | Static descriptor MID info as configured on the gateway |
| amount | integer | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| sent_to_acquirer | string(255) | "true" or "false" |

---

## Required vs Optional API params

There are some API params which can be configured as either optional or required on the terminal level. Contact the IT Support team at tech-support@e-comprocessing.com if you wish to disable or enable some API params based on your business model and integration type. Note also that if all the billing address params are configured as optional, then the whole billing address XML tag can be dismissed and not sent in the API requests.

| Name | Type | Description |
|---|---|---|
| customer_email | email address | By default, customer email is a required API param |
| customer_phone | string(32) | By default, customer phone is an optional API param |
| remote_ip | string(40) | By default, customer IP is a required API param |
| first_name | string(255) | By default, customer first name is a required API param |
| last_name | string(255) | By default, customer last name is a required API param |
| address1 | string(255) | By default, primary address is a required API param. Cannot be configured as optional for account verification transaction type |
| address2 | string(255) | By default, secondary address is an optional API param |
| city | string(255) | By default, city is a required API param. Cannot be config- ured as optional for account verification transaction type |
| zip_code | string | By default, zip code is a required API param. Cannot be con- figured as optional for account verification transaction type |
| state | string(2) | By default, state is an optional API param (unless country is US or CA) |
| country | string(2) | By default, country code is a required API param |
| usage | string(255) | By default, usage will be configured as a required API param for SDD and P24, optional for the rest |

---

## Transaction States

Transactions will have one of the following states. These will be returned by transaction responses, reconcile responses and will be shown in the browser interface

| Status | Description |
|---|---|
| approved | Transaction was approved by the schemes and is successful. |
| declined | Transaction was declined by the schemes or risk management. |
| pending_async | An asynchronous transaction (3-D secure payment) has been initiated and is waiting for user input. Updates of this state will be sent to the notification url specified in request. |
| pending | The outcome of the transaction could not be determined, e.g. at a timeout situation. Transaction state will eventually change, so make a reconcile after a certain time frame. |
| error | An error has occurred while negotiating with the schemes. |
| refunded | Once an approved transaction is refunded the state changes to refunded. |

| Status | Description |
|---|---|
| chargebacked | Once an approved transaction is chargebacked - the state changes to changebacked. Chargeback is the state of rejecting an accepted transaction (with funds transferred) by the cardholder or the issuer |
| voided | Transaction was authorized, but later the merchant canceled it. |
| chargeback_reversed | Once a chargebacked transaction is charged, the state changes to chargeback reversed. Chargeback has been canceled. |
| represented | Once a chargebacked transaction is charged, the state changes to represented. Chargeback has been canceled. |
| second_chargebacked | Once a chargeback_reversed/represented transaction is chargebacked the state changes to second chargebacked. |
| pending_review | Transaction on hold, a manual review will be done |

## Supported Card Brands

Card Brands are specific per acquirer. If you want to use a specific card brand you can contactech-support@e-comprocessing.com.

| Supported card brands |
|---|
| Visa |
| Master |
| Intl Maestro |
| Discover |
| Diners |
| AMEX |
| JCB |
| RuPay |
| Elo |
| Aura |
| Hipercard |

## Document ID Parameter

Document ID is consumer personal identification. It is different for every country and is described more specifically in the table below. Document ID is required for some of the acquirers, please contact the IT Support team at tech-support@e-comprocessing.com for more information.

| Country | Name | Type | Description |
|---|---|---|---|
| Argentina | document_id | string(255) | Consumer's Argentinian Identification Number(DNI or CUIT). Must be string between 7 to 9, or 11 digits. |
| Brazil | document_id | string(255) | Consumer's Brazilian Identification Number(CPF or CNPJ). Must be string between 11 and 14 digits and to have full cpf validation. Example: 76484475687 |
| Chile | document_id | string(255) | Consumer's Chilean Identification Number(CI/RUT). Must be string between 8 to 9 digits. |
| Colombia | document_id | string(255) | Consumer's Colombian Identification Number(CC). Must be string between 6 to 10 digits. |
| India | document_id | string(255) | Consumer's Indian PAN. Must be string with 10 alphanumeric letters. 5 letters, followed by 4 numbers, followed by 1 letter or number. Example: ABCDE1234F |
| Mexico | document_id | string(255) | Consumer's Mexican Identification Number(CURP). Must be string between 10 and 18 digits. |
| Paraguay | document_id | string(255) | Consumer's Paraguayan Identification Number(CI). Must be string between 5 and 20 digits. |
| Peru | document_id | string(255) | Consumer's Peruvian Identification Number(DNI). Must be string between 8 and 9 digits. |
| Turkey | document_id | string(255) | Consumer's Turkish Identification Number(T.C. Kimlik No.). Must be string between 5 and 20 digits. |
| Uruguay | document_id | string(255) | Consumer's Uruguayan Identification Number(CI). Must be string between 6 and 8 digits. |

## Business Attributes

Business attributes are groups of additional risk attributes which are in close relation with the merchant business category. Some/All of them can be required at our risk team's discretion and

will be used for internal reporting only. These business attributes can be submitted with a standard card transaction on Processing API and WPF processing.

**Business categories:**

| Segment | MCC |
|---|---|
| **Airlines Air Carriers** | |
| Airlines, Air Carriers | 4511 |
| Airlines | 3000 - 3299 |
| **Event Management** | |
| Consulting, Public Relations | 7392 |
| Miscellaneous General Services | 7299 |
| Theatrical Ticket Agencies | 7922 |
| Direct Marketing - Other | 5969 |
| **Furniture** | |
| Furniture, Home Furnishings, and Equipment Stores, Except Appliances | 5712 |
| Office and Commercial Furniture | 5021 |
| **Hotels and Real estate Rentals** | |
| Hotels/Motels/Inns/Resorts | 3501 - 3790 |
| Real Estate Agents and Managers - Rentals | 6513 |
| Lodging – Hotels, Motels, Resorts, Central Reservation Services (not elsewhere classified) | 7011 |
| Timeshares | 7012 |
| **Car, plane and Boat rentals** | |
| Car Rental | 3351 - 3441 |
| Taxicabs and Limousines | 4121 |
| Bus Lines, Including Charters, Tour Buses | 4131 |
| Boat Rentals and Leases | 4457 |
| Transportation Services, (Not elsewhere classified) | 4789 |
| Car Rental Companies | 7512 |
| Truck and Utility Trailer Rentals | 7513 |
| Motor Home and Recreational Vehicle Rentals | 7519 |
| **Cruise Lines** | |
| Cruise Lines | 4411 |
| **Travel Agencies** | |
| Travel Agencies | 4722 |
| Package Tour Operators (For use in Germany only) | 4723 |
| Direct Marketing - Travel-related Arrangement Services | 5962 |

Specific attributes for each business category can be found in the following table:

| Attribute | Type | Description |
|---|---|---|
| **Airlines Air Carriers** | | |
| flight_arrival_date | string | The date when the flight arrives in format dd-mm-yyyy |
| flight_departure_date | string | The date when the flight departs in format dd-mm-yyyy |
| airline_code | string | The code of Airline |
| airline_flight_number | string | The flight number |
| flight_ticket_number | string | The number of the flight ticket |
| flight_origin_city | string | The origin city of the flight |
| flight_destination_city | string | The destination city of the flight |

| Attribute | Type | Description |
|---|---|---|
| airline_tour_operator_name | string | The name of tour operator |
| **Event Management** | | |
| event_start_date | string | The date when event starts in format dd-mm-yyyy |
| event_end_date | string | The date when event ends in format dd-mm-yyyy |
| event_organizer_id | string | |
| event_id | string | |
| **Furniture** | | |
| date_of_order | string | The date when order was placed in format dd-mm-yyyy |
| delivery_date | string | Date of the expected delivery in format dd-mm-yyyy |
| name_of_the_supplier | string | |
| **Hotels and Real estate rentals** | | |
| check_in_date | string | The data when the customer check-in in format dd-mm-yyyy |
| check_out_date | string | The data when the customer check-out in format dd-mm-yyyy |
| travel_agency_name | string | |
| **Car, Plane and Boat Rentals** | | |
| vehicle_pick_up_date | string | The date when customer takes the vehicle in format dd-mm-yyyy |
| vehicle_return_date | string | The date when the customer returns the vehicle back in format dd-mm-yyyy |
| supplier_name | string | The name of supplier/contractor |
| **Cruise Lines** | | |
| cruise_start_date | string | The date when cruise begins in format dd-mm-yyyy |
| cruise_end_date | string | The date when cruise ends in format dd-mm-yyyy |
| **Travel Agencies** | | |
| arrival_date | string | The date of arrival in format dd-mm-yyyy |
| departure_date | string | The date of departure in format dd-mm-yyyy |
| carrier_code | string | The code of the carrier |
| flight_number | string | The number of the flight |
| ticket_number | string | The number of the ticket |
| origin_city | string | The origin city |
| destination_city | string | The destination city |
| travel_agency | string | The name of the travel agency |
| contractor_name | string | The name of the contractor |
| atol_certificate | string | ATOL certificate number |
| pick_up_date | string | Pick-up date in format dd-mm-yyyy |
| return_date | string | Return date in format dd-mm-yyyy |

**Transaction types with business attributes:**

- authorize
- authorize3d
- capture
- sale
- sale3d
- init_recurring_sale
- init_recurring_sale3d
- recurring_sale

# Reconcile

Reconcile can be used to retrieve data about a transaction. This can be useful if you want to retrieve information about a transaction whose status is timeout, which returned an error or has changed eg. has beed chargebacked.

Reconcile request are handled exactly like transaction requests via XML.

---

## Single Transaction

The URL for single transaction reconciling is similar to the processing url:

`https://username:password@staging.gate.e-comprocessing.net/reconcile/TERMINAL-TOKEN`

Reconcile By Unique Id Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('NonFinancial\Reconcile\Transaction');
    $request = $genesis->request();

    $request
        ->setUniqueId('44177a21403427eb96664a6d7e5d5d48');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.nonfinancial.reconcile.ReconcileRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    ReconcileRequest request = new ReconcileRequest();

    request.setUniqueId("44177a21403427eb96664a6d7e5d5d48");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.reconcile(
  {
    "unique_id": "44177a21403427eb96664a6d7e5d5d48"
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/reconcile/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <reconcile>
    <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  </reconcile>'
```

Reconcile By Arn Request

```
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('NonFinancial\Reconcile\Transaction');
    $request = $genesis->request();

    $request
        ->setArn('74537605248535042582882');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.nonfinancial.reconcile.ReconcileRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    ReconcileRequest request = new ReconcileRequest();

    request.setArn("74537605248535042582882");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.reconcile(
  {
    "arn": "74537605248535042582882"
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/reconcile/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <reconcile>
    <arn>74537605248535042582882</arn>
  </reconcile>'
```

Reconcile By Transaction Id Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('NonFinancial\Reconcile\Transaction');
    $request = $genesis->request();

    $request
        ->setTransactionId('merchant-transaction-id-here');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.nonfinancial.reconcile.ReconcileRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    ReconcileRequest request = new ReconcileRequest();

    request.setTransactionId("merchant-transaction-id-here");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.reconcile(
  {
    "transaction_id": "merchant-transaction-id-here"
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/reconcile/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <reconcile>
    <transaction_id>merchant-transaction-id-here</transaction_id>
  </reconcile>'
```

**XML Request to reconcile:**

Note that reconcile can be done via either unique_id, ARN or transaction_id

**XML Response:**

Response is a standard payment response like it would be returned by any transaction. It can have either state as shown in the states section.

Successful Sale Transaction Reconciliation Response

```
stdClass Object
(
    [transaction_type] => sale
    [status] => approved
    [authorization_code] => 005645
    [retrieval_reference_number] => 016813015184
    [response_code] => 00
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [transaction_id] => 119643250547501c79d8295
    [mode] => live
    [timestamp] => 2020-07-15T12:08:24Z
    [descriptor] => Descriptor one
    [amount] => 9000
    [currency] => USD
    [card_brand] => visa
    [card_number] => 420000...0000
    [card_type] => CREDIT
    [card_subtype] => CARD SUBTYPE
    [card_issuing_bank] => Issuing Bank
    [bank_account_number] => Bank Account Number
    [bank_identifier_code] => Bank Identifier Code
    [sent_to_acquirer] => true
    [arn] => 74537605259536043849425
)
```

```
<payment_response content=[
    <transaction_type content=[sale]>
    <status content=[approved]>
    <authorization_code content=[005645]>
    <retrieval_reference_number content=[016813015184]>
    <response_code content=[00]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <transaction_id content=[119643250547501c79d8295]>
    <mode content=[live]>
    <timestamp content=[2020-07-15T12:08:24Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[9000]>
    <currency content=[USD]>
    <card_brand content=[visa]>
    <card_number content=[420000...0000]>
    <card_type content=[CREDIT]>
    <card_subtype content=[CARD SUBTYPE]>
    <card_issuing_bank content=[Issuing Bank]>
    <bank_account_number content=[Bank Account Number]>
    <bank_identifier_code content=[Bank Identifier Code]>
    <sent_to_acquirer content=[true]>
    <arn content=[74537605259536043849425]>
]>
```

```
{
    transaction_type: "sale",
    status: "approved",
    authorization_code: "005645",
    retrieval_reference_number: "016813015184",
    response_code: "00",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    transaction_id: "119643250547501c79d8295",
    mode: "live",
    timestamp: "2020-07-15T12:08:24Z",
    descriptor: "Descriptor one",
    amount: "9000",
    currency: "USD",
    card_brand: "visa",
    card_number: "420000...0000",
    card_type: "CREDIT",
    card_subtype: "CARD SUBTYPE",
    card_issuing_bank: "Issuing Bank",
    bank_account_number: "Bank Account Number",
    bank_identifier_code: "Bank Identifier Code",
    sent_to_acquirer: "true",
    arn: "74537605259536043849425",
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_response>
  <transaction_type>sale</transaction_type>
  <status>approved</status>
  <authorization_code>005645</authorization_code>
  <retrieval_reference_number>016813015184</retrieval_reference_number>
  <response_code>00</response_code>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <transaction_id>119643250547501c79d8295</transaction_id>
  <mode>live</mode>
  <timestamp>2020-07-15T12:08:24Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>9000</amount>
  <currency>USD</currency>
  <card_brand>visa</card_brand>
  <card_number>420000...0000</card_number>
  <card_type>CREDIT</card_type>
  <card_subtype>CARD SUBTYPE</card_subtype>
  <card_issuing_bank>Issuing Bank</card_issuing_bank>
  <bank_account_number>Bank Account Number</bank_account_number>
  <bank_identifier_code>Bank Identifier Code</bank_identifier_code>
  <sent_to_acquirer>true</sent_to_acquirer>
  <arn>74537605259536043849425</arn>
</payment_response>
```


Card brand and card number will be available in response only for card transaction types.

The reference transaction unique id is also returned when a reference-based transaction has been queried via the Reconcile API.

Successful Refund Transaction Reconciliation Response


```
stdClass Object
(
    [transaction_type] => refund
    [status] => approved
    [authorization_code] => 005645
    [retrieval_reference_number] => 016813015184
    [response_code] => 00
    [unique_id] => 5de39380bf7ac7e1fc31cbd7805dc0ec
    [transaction_id] => 119643250547501c79d8295
    [mode] => live
    [timestamp] => 2020-07-15T12:08:24Z
    [descriptor] => Descriptor one
    [amount] => 9000
    [currency] => USD
    [card_brand] => visa
    [card_number] => 420000...0000
    [card_type] => CREDIT
    [card_subtype] => CARD SUBTYPE
    [card_issuing_bank] => Issuing Bank
    [bank_account_number] => Bank Account Number
    [bank_identifier_code] => Bank Identifier Code
    [sent_to_acquirer] => true
    [arn] => 74537605259536043849425
    [reference_transaction_unique_id] => 44177a21403427eb96664a6d7e5d5d48
)
```



```
<payment_response content=[
    <transaction_type content=[refund]>
    <status content=[approved]>
    <authorization_code content=[005645]>
    <retrieval_reference_number content=[016813015184]>
    <response_code content=[00]>
    <unique_id content=[5de39380bf7ac7e1fc31cbd7805dc0ec]>
    <transaction_id content=[119643250547501c79d8295]>
    <mode content=[live]>
    <timestamp content=[2020-07-15T12:08:24Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[9000]>
    <currency content=[USD]>
    <card_brand content=[visa]>
    <card_number content=[420000...0000]>
    <card_type content=[CREDIT]>
    <card_subtype content=[CARD SUBTYPE]>
    <card_issuing_bank content=[Issuing Bank]>
    <bank_account_number content=[Bank Account Number]>
    <bank_identifier_code content=[Bank Identifier Code]>
    <sent_to_acquirer content=[true]>
    <arn content=[74537605259536043849425]>
    <reference_transaction_unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
]>
```

```
{
    transaction_type: "refund",
    status: "approved",
    authorization_code: "005645",
    retrieval_reference_number: "016813015184",
    response_code: "00",
    unique_id: "5de39380bf7ac7e1fc31cbd7805dc0ec",
    transaction_id: "119643250547501c79d8295",
    mode: "live",
    timestamp: "2020-07-15T12:08:24Z",
    descriptor: "Descriptor one",
    amount: "9000",
    currency: "USD",
    card_brand: "visa",
    card_number: "420000...0000",
    card_type: "CREDIT",
    card_subtype: "CARD SUBTYPE",
    card_issuing_bank: "Issuing Bank",
    bank_account_number: "Bank Account Number",
    bank_identifier_code: "Bank Identifier Code",
    sent_to_acquirer: "true",
    arn: "74537605259536043849425",
    reference_transaction_unique_id: "44177a21403427eb96664a6d7e5d5d48",
}
```



```
<?xml version="1.0" encoding="UTF-8"?>
<payment_response>
  <transaction_type>refund</transaction_type>
  <status>approved</status>
  <authorization_code>005645</authorization_code>
  <retrieval_reference_number>016813015184</retrieval_reference_number>
  <response_code>00</response_code>
  <unique_id>5de39380bf7ac7e1fc31cbd7805dc0ec</unique_id>
  <transaction_id>119643250547501c79d8295</transaction_id>
  <mode>live</mode>
  <timestamp>2020-07-15T12:08:24Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>9000</amount>
  <currency>USD</currency>
  <card_brand>visa</card_brand>
  <card_number>420000...0000</card_number>
  <card_type>CREDIT</card_type>
  <card_subtype>CARD SUBTYPE</card_subtype>
  <card_issuing_bank>Issuing Bank</card_issuing_bank>
  <bank_account_number>Bank Account Number</bank_account_number>
  <bank_identifier_code>Bank Identifier Code</bank_identifier_code>
  <sent_to_acquirer>true</sent_to_acquirer>
  <arn>74537605259536043849425</arn>
  <reference_transaction_unique_id>44177a21403427eb96664a6d7e5d5d48</reference_transaction_unique_id>
</payment_response>
```


Captured flag and possible remaining capturable amount would be returned when an authorization transaction has been queried via the Reconcile API.

Successful Authorize Transaction Reconciliation Response


```
stdClass Object
(
    [transaction_type] => authorize
    [status] => approved
    [authorization_code] => 005645
    [retrieval_reference_number] => 016813015184
    [response_code] => 00
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [transaction_id] => 119643250547501c79d8295
    [mode] => live
    [timestamp] => 2020-07-15T12:08:24Z
    [descriptor] => Descriptor one
    [amount] => 500
    [currency] => USD
    [card_brand] => visa
    [card_number] => 420000...0000
    [card_type] => CREDIT
    [card_subtype] => CARD SUBTYPE
    [card_issuing_bank] => Issuing Bank
    [bank_account_number] => Bank Account Number
    [bank_identifier_code] => Bank Identifier Code
    [sent_to_acquirer] => true
    [arn] => 74537605259536043849425
    [captured] => false
    [capturable_amount] => 500
)
```

```xml
<payment_response content=[
    <transaction_type content=[authorize]>
    <status content=[approved]>
    <authorization_code content=[005645]>
    <retrieval_reference_number content=[016813015184]>
    <response_code content=[00]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <transaction_id content=[119643250547501c79d8295]>
    <mode content=[live]>
    <timestamp content=[2020-07-15T12:08:24Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[500]>
    <currency content=[USD]>
    <card_brand content=[visa]>
    <card_number content=[420000...0000]>
    <card_type content=[CREDIT]>
    <card_subtype content=[CARD SUBTYPE]>
    <card_issuing_bank content=[Issuing Bank]>
    <bank_account_number content=[Bank Account Number]>
    <bank_identifier_code content=[Bank Identifier Code]>
    <sent_to_acquirer content=[true]>
    <arn content=[74537605259536043849425]>
    <captured content=[false]>
    <capturable_amount content=[500]>
]>
```

```json
{
    transaction_type: "authorize",
    status: "approved",
    authorization_code: "005645",
    retrieval_reference_number: "016813015184",
    response_code: "00",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    transaction_id: "119643250547501c79d8295",
    mode: "live",
    timestamp: "2020-07-15T12:08:24Z",
    descriptor: "Descriptor one",
    amount: "500",
    currency: "USD",
    card_brand: "visa",
    card_number: "420000...0000",
    card_type: "CREDIT",
    card_subtype: "CARD SUBTYPE",
    card_issuing_bank: "Issuing Bank",
    bank_account_number: "Bank Account Number",
    bank_identifier_code: "Bank Identifier Code",
    sent_to_acquirer: "true",
    arn: "74537605259536043849425",
    captured: "false",
    capturable_amount: "500",
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_response>
  <transaction_type>authorize</transaction_type>
  <status>approved</status>
  <authorization_code>005645</authorization_code>
  <retrieval_reference_number>016813015184</retrieval_reference_number>
  <response_code>00</response_code>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <transaction_id>119643250547501c79d8295</transaction_id>
  <mode>live</mode>
  <timestamp>2020-07-15T12:08:24Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>500</amount>
  <currency>USD</currency>
  <card_brand>visa</card_brand>
  <card_number>420000...0000</card_number>
  <card_type>CREDIT</card_type>
  <card_subtype>CARD SUBTYPE</card_subtype>
  <card_issuing_bank>Issuing Bank</card_issuing_bank>
  <bank_account_number>Bank Account Number</bank_account_number>
  <bank_identifier_code>Bank Identifier Code</bank_identifier_code>
  <sent_to_acquirer>true</sent_to_acquirer>
  <arn>74537605259536043849425</arn>
  <captured>false</captured>
  <capturable_amount>500</capturable_amount>
</payment_response>
```


**PREAUTHORIZATION**

Custom response data will be returned when aPreauthorization transaction has been queried via the Reconcile API.

| Name | Type | Description |
|---|---|---|
| preauthorization | "true" | Preauthorization flag |
| preauthorization_expires_at | string | Preauthorization expiration date time inISO 8601 Combined date and time, e.g. 2007-08-30T17:46:11Z |

| Name | Type | Description |
|---|---|---|
| preauthorization_total_amount | integer | Total preauthorization amount (initial + topup amount) |
| capturable_amount | integer | The total amount that can be captured |
| captured_amount | integer | The total captured amount |
| reversed_amount | integer | The total reversed amount |
| reversible_amount | integer | The total reversible amount |

The total capturable amount will be decreased in case there is/are already submitted partial reversal(s).

Successful Preauthorization Reconciliation Response

```
stdClass Object
(
    [transaction_type] => authorize
    [status] => approved
    [authorization_code] => 005645
    [retrieval_reference_number] => 016813015184
    [response_code] => 00
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [transaction_id] => 119643250547501c79d8295
    [mode] => live
    [timestamp] => 2020-07-15T12:08:24Z
    [descriptor] => Descriptor one
    [amount] => 5000
    [currency] => USD
    [card_brand] => master
    [card_number] => 555555...4444
    [card_type] => CREDIT
    [card_subtype] => CARD SUBTYPE
    [card_issuing_bank] => Issuing Bank
    [bank_account_number] => Bank Account Number
    [bank_identifier_code] => Bank Identifier Code
    [sent_to_acquirer] => true
    [arn] => 74537605259536043849425
    [preauthorization] => true
    [preauthorization_expires_at] => 2020-08-15T15:08:24+03:00
    [preauthorization_total_amount] => 10000
    [captured] => false
    [capturable_amount] => 8000
    [captured_amount] => 0
    [reversed_amount] => 2000
    [reversible_amount] => 8000
)
```

```
<payment_response content=[
    <transaction_type content=[authorize]>
    <status content=[approved]>
    <authorization_code content=[005645]>
    <retrieval_reference_number content=[016813015184]>
    <response_code content=[00]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <transaction_id content=[119643250547501c79d8295]>
    <mode content=[live]>
    <timestamp content=[2020-07-15T12:08:24Z]>
    <descriptor content=[Descriptor one]>
    <amount content=[5000]>
    <currency content=[USD]>
    <card_brand content=[master]>
    <card_number content=[555555...4444]>
    <card_type content=[CREDIT]>
    <card_subtype content=[CARD SUBTYPE]>
    <card_issuing_bank content=[Issuing Bank]>
    <bank_account_number content=[Bank Account Number]>
    <bank_identifier_code content=[Bank Identifier Code]>
    <sent_to_acquirer content=[true]>
    <arn content=[74537605259536043849425]>
    <preauthorization content=[true]>
    <preauthorization_expires_at content=[2020-08-15T15:08:24+03:00]>
    <preauthorization_total_amount content=[10000]>
    <captured content=[false]>
    <capturable_amount content=[8000]>
    <captured_amount content=[0]>
    <reversed_amount content=[2000]>
    <reversible_amount content=[8000]>
]>
```

```
{
    transaction_type: "authorize",
    status: "approved",
    authorization_code: "005645",
    retrieval_reference_number: "016813015184",
    response_code: "00",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    transaction_id: "119643250547501c79d8295",
    mode: "live",
    timestamp: "2020-07-15T12:08:24Z",
    descriptor: "Descriptor one",
    amount: "5000",
    currency: "USD",
    card_brand: "master",
    card_number: "555555...4444",
    card_type: "CREDIT",
    card_subtype: "CARD SUBTYPE",
    card_issuing_bank: "Issuing Bank",
    bank_account_number: "Bank Account Number",
    bank_identifier_code: "Bank Identifier Code",
    sent_to_acquirer: "true",
    arn: "74537605259536043849425",
    preauthorization: "true",
    preauthorization_expires_at: "2020-08-15T15:08:24+03:00",
    preauthorization_total_amount: "10000",
    captured: "false",
    capturable_amount: "8000",
    captured_amount: "0",
    reversed_amount: "2000",
    reversible_amount: "8000",
}
```



```
<?xml version="1.0" encoding="UTF-8"?>
<payment_response>
  <transaction_type>authorize</transaction_type>
  <status>approved</status>
  <authorization_code>005645</authorization_code>
  <retrieval_reference_number>016813015184</retrieval_reference_number>
  <response_code>00</response_code>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <transaction_id>119643250547501c79d8295</transaction_id>
  <mode>live</mode>
  <timestamp>2020-07-15T12:08:24Z</timestamp>
  <descriptor>Descriptor one</descriptor>
  <amount>5000</amount>
  <currency>USD</currency>
  <card_brand>master</card_brand>
  <card_number>555555...4444</card_number>
  <card_type>CREDIT</card_type>
  <card_subtype>CARD SUBTYPE</card_subtype>
  <card_issuing_bank>Issuing Bank</card_issuing_bank>
  <bank_account_number>Bank Account Number</bank_account_number>
  <bank_identifier_code>Bank Identifier Code</bank_identifier_code>
  <sent_to_acquirer>true</sent_to_acquirer>
  <arn>74537605259536043849425</arn>
  <preauthorization>true</preauthorization>
  <preauthorization_expires_at>2020-08-15T15:08:24+03:00</preauthorization_expires_at>
  <preauthorization_total_amount>10000</preauthorization_total_amount>
  <captured>false</captured>
  <capturable_amount>8000</capturable_amount>
  <captured_amount>0</captured_amount>
  <reversed_amount>2000</reversed_amount>
  <reversible_amount>8000</reversible_amount>
</payment_response>
```


---

# By date range

Date range based reconciliation allows you to fetch information for all payment transactions from a terminal within a given date range. The response is paginated, each request will return 100 entries max.

The URL for date range reconciling is:

`https://username:password@staging.gate.e-comprocessing.net/reconcile/by_date/TERMINAL-TOKEN`

Reconcile By Date Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('NonFinancial\Reconcile\DateRange');
    $request = $genesis->request();

    $request
        ->setStartDate('2014-01-01 09:20:00')
        ->setEndDate('2014-01-31 21:30:00')
        ->setPage('2');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.nonfinancial.reconcile.ReconcileByDate;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    ReconcileByDate request = new ReconcileByDate();

    request.setStartDate("2014-01-01 09:20:00");
    request.setEndDate("2014-01-31 21:30:00");
    request.setPage("2");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```node
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.reconcile_by_date(
  {
    "start_date": "2014-01-01 09:20:00",
    "end_date": "2014-01-31 21:30:00",
    "page": 2
  }
).send()
  .then(success)
  .catch(failure);
```

```curl
curl https://username:password@staging.gate.e-comprocessing.net/reconcile/by_date/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <reconcile>
    <start_date>2014-01-01 09:20:00</start_date>
    <end_date>2014-01-31 21:30:00</end_date>
    <page>2</page>
  </reconcile>'
```

| Parameter | Required | Format | Description |
|---|---|---|---|
| start_date | required | yyyy-mm-dd hh:mm:ss | start of the requested date range (time is optional) |
| end_date | optional | yyyy-mm-dd hh:mm:ss | end of the requested date range (time is optional) |
| page | optional | integer the page within the paginated result, defaults to 1 | |

**Response:**

The attributes in the root node payment responses include information about the pagination of the response.


```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_responses per_page="100" page="2" total_count="19" pages_count="7">
  <payment_response>
    <transaction_type>sale</transaction_type>
    <status>approved</status>
    <authorization_code>005645</authorization_code>
    <response_code>00</response_code>
    <unique_id>130319cfb3bf65ff3c4a4045487b174f</unique_id>
    <transaction_id>EFBFBC7D-82CD-4375-9A69-15F19C88A134</transaction_id>
    <technical_message>Transaction successful!</technical_message>
    <message>Transaction successful!</message>
    <mode>live</mode>
    <timestamp>2014-01-03T15:04:00Z</timestamp>
    <descriptor>descriptor one</descriptor>
    <amount>500</amount>
    <currency>USD</currency>
    <card_brand>visa</card_brand>
    <card_number>420000...0000</card_number>
    <card_holder>John Doe</card_holder>
    <expiration_year>2020</expiration_year>
    <expiration_month>2</expiration_month>
    <sent_to_acquirer>true</sent_to_acquirer>
  </payment_response>
  <payment_response>
    <transaction_type>sale</transaction_type>
    <status>approved</status>
    <authorization_code>638745</authorization_code>
    <response_code>00</response_code>
    <unique_id>130319cfb3bf65ff3c4a4045487b173f</unique_id>
    <transaction_id>BBD7945B-BE57-4A14-A7FB-47F7AE928D95</transaction_id>
    <technical_message>Transaction successful!</technical_message>
    <message>Transaction successful!</message>
    <mode>live</mode>
    <timestamp>2014-01-05T15:04:00Z</timestamp>
    <descriptor>descriptor one</descriptor>
    <amount>500</amount>
    <currency>USD</currency>
    <card_brand>visa</card_brand>
    <card_number>420000...0000</card_number>
    <card_holder>John Doe</card_holder>
    <expiration_year>2020</expiration_year>
    <expiration_month>2</expiration_month>
    <sent_to_acquirer>true</sent_to_acquirer>
  </payment_response>
  <payment_response>
    <transaction_type>sale</transaction_type>
    <status>approved</status>
    <authorization_code>226534</authorization_code>
    <response_code>00</response_code>
    <unique_id>1e8a6f09253eb8b4f8c84c0d8803713e</unique_id>
    <transaction_id>_5041_2013041012_22_10_545</transaction_id>
    <technical_message>Transaction successful!</technical_message>
    <message>Transaction successful!</message>
    <mode>live</mode>
    <timestamp>2013-01-09T10:22:13Z</timestamp>
    <descriptor>test_descriptor</descriptor>
    <amount>50422</amount>
    <currency>EUR</currency>
    <card_brand>visa</card_brand>
    <card_number>420000...0000</card_number>
    <card_holder>John Doe</card_holder>
    <expiration_year>2020</expiration_year>
    <expiration_month>2</expiration_month>
    <sent_to_acquirer>true</sent_to_acquirer>
  </payment_response>
  <payment_response>
    <transaction_type>sale3d</transaction_type>
    <status>pending_async</status>
    <unique_id>5dbdb4c677ee16b8fb1e43483164be2c</unique_id>
    <transaction_id>_6547_2013041012_23_08_470</transaction_id>
    <technical_message>Transaction successful!</technical_message>
    <message>Transaction successful!</message>
    <redirect_url>https://staging.gate.e-comprocessing.net/redirect/to_acquirer/5dbdb4c677</redirect_url>
    <mode>live</mode>
    <timestamp>2014-01-10T10:23:10Z</timestamp>
    <descriptor>test_descriptor</descriptor>
    <amount>100</amount>
    <currency>EUR</currency>
    <card_brand>visa</card_brand>
    <card_number>471110...0000</card_number>
    <card_holder>John Doe</card_holder>
    <expiration_year>2020</expiration_year>
    <expiration_month>2</expiration_month>
```

```
    <sent_to_acquirer>true</sent_to_acquirer>
  </payment_response>
  <payment_response>
    <transaction_type>refund</transaction_type>
    <status>approved</status>
    <authorization_code>005645</authorization_code>
    <response_code>00</response_code>
    <unique_id>44177a21403427eb96664a6d7e5d5d49</unique_id>
    <transaction_id>119643250547501c79d8296</transaction_id>
    <technical_message>Transaction successful!</technical_message>
    <message>Transaction successful!</message>
    <mode>test</mode>
    <timestamp>2014-01-30T14:21:48Z</timestamp>
    <descriptor>descriptor one</descriptor>
    <amount>9000</amount>
    <currency>USD</currency>
    <sent_to_acquirer>true</sent_to_acquirer>
    <reference_transaction_unique_id>44177a21403427eb96664a6d7e5d5d48</reference_transaction_unique_id>
   </payment_response>
  ...
</payment_responses>
```

| Name | Type | Description |
|---|---|---|
| @per_page | integer | number of entries per page |
| @page | integer | the current page |
| @total_count | integer | total number of all entries |
| @pages_count | integer | total number of pages |

# Payment Authorizations

#### PAYMENT AUTHORIZATIONS API

The Payment Authorizations API can be used to retrieve data about payment authorizations.

Payment Authorizations can be retrieved by date

#### BY DATE RANGE

Date range based retrieval allows you to fetch information for all payment authorizations for a given merchant within a given date range. Date range searches for payment authorizations by their creation date. The response is paginated, each request will return 100 entries max.

The URLs for date range payment authorizations retrieval are:

Production:

`https://gate.e-comprocessing.net/payment_authorizations/by_date`

Staging (for integration):

`https://staging.gate.e-comprocessing.net/payment_authorizations/by_date`

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://staging.gate.e-comprocessing.net/payment_authorizations/by_date \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_authorization_request>
    <start_date>2014-01-01</start_date>
    <end_date>2014-01-31</end_date>
    <auth_start_date>2014-01-01</auth_start_date>
    <auth_end_date>2014-01-31</auth_end_date>
    <externally_processed>external</externally_processed>
    <processing_type>all</processing_type>
    <page>1</page>
  </payment_authorization_request>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|

| Parameter | Required | Format | Description |
|---|---|---|---|
| start_date | required* | yyyy-mm-dd | start of the requested date range |
| end_date | optional | yyyy-mm-dd | end of the requested date range |
| auth_start_date | required* | yyyy-mm-dd | start of the requested auth date range |
| auth_end_date | optional | yyyy-mm-dd | end of the requested auth date range |
| page | optional | integer | the page within the paginated result, defaults to 1 |
| per_page | optional | integer | Number of entities on page, defaults to 100 |
| externally_processed | optional | string(255) | Filters transactions by being externally processed: or being native to Genesis. Possible values include 'genesis', 'external', and 'all'. If flag not supplied, it defaults to 'genesis' |
| processing_type | optional | string(255) | Filters transactions by being card present or card not present. Possible values include 'card_present', 'card_not_present', and 'all'. If flag not supplied, it defaults to 'all'. |

`required* = conditionally required`

Note: One of **start_date** and **auth_start_date** is required.

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| merchant_number | string(20) | Merchant number |
| record_number | string | Unique identifier for the authorizations |
| card_number | string(19) | Masked card number |
| exp_date | string(4) | Expiry MMYY |
| currency | string(3) | Currency |
| amount | integer | Amount in minor currency unit, seeCurrency and Amount Handling for details |
| auth_code | string(6) | Authorization code |
| auth_date | string | Authorization date in date and time format |
| resp_code | string | Description of the response code |
| reversed | string(1) | Contains the character Y if the authorisation has been reversed else N |
| pos_entry_mode | string(3) | POS Entry Mode |
| voice | string(1) | Y if this is a voice authorisation, N otherwise |
| avs_result | string(1) | AVS Result |
| cvv2_result | string(1) | CVV2 Result |
| card_type | string(1) | Card Type code |
| ecom_type | string(5) | Indicator for electronic commerce transactions |
| eci_sli | string(2) | Visa card scheme ECI / MasterCard SLI |
| rrn | string(12) | Retrieval Reference Number |
| card_acceptor | string(40) | Card Acceptor Name and Location |
| mcc | string(4) | Merchant category code |
| trace | integer | Trace number. All messages belonging the same transaction have to have the same trace value |
| type | string(1) | Transaction Type |
| card_sub_type | string(8) | Card Subtype |
| terminal_id | string(15) | Terminal ID |
| unique_tran_id | string(36) | Unique Transaction Identifier |

Successful Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```


```xml
<?xml version="1.0" encoding="UTF-8"?>
<payment_authorizations_responses per_page="100" page="1" total_count="2" pages_count="1">
  <payment_authorizations_response>
    <xml_root>payment_authorization_response</xml_root>
    <merchant_number>605000000002029</merchant_number>
    <record_number>33202414090</record_number>
    <card_number>465944******7359</card_number>
    <exp_date/>
    <currency>EUR</currency>
    <amount>1234</amount>
    <auth_code>15056</auth_code>
    <auth_date>2019-06-10 16:17:18 UTC</auth_date>
    <resp_code>Transaction approved</resp_code>
    <reversed>N</reversed>
    <pos_entry_mode>102</pos_entry_mode>
    <voice>N</voice>
    <avs_result/>
    <cvv2_result>M</cvv2_result>
    <card_type>M</card_type>
    <ecom_type/>
    <eci_sli>21</eci_sli>
    <rrn>916116130233</rrn>
    <card_acceptor>test.bg                12629314     BG</card_acceptor>
    <mcc>7995</mcc>
    <trace>1130233</trace>
    <type/>
    <card_sub_type/>
    <terminal_id>53b1f5eacc9edd</terminal_id>
    <unique_tran_id/>
  </payment_authorizations_response>
  <payment_authorizations_response>
    <merchant_number>605000000002029</merchant_number>
    <record_number>33202414090</record_number>
    <card_number>465944******7359</card_number>
    <exp_date/>
    <currency>EUR</currency>
    <amount>1234</amount>
    <auth_code>15056</auth_code>
    <auth_date>2019-06-10 16:17:18 UTC</auth_date>
    <resp_code>Transaction approved</resp_code>
    <reversed>N</reversed>
    <pos_entry_mode>102</pos_entry_mode>
    <voice>N</voice>
    <avs_result/>
    <cvv2_result>M</cvv2_result>
    <card_type>M</card_type>
    <ecom_type/>
    <eci_sli>21</eci_sli>
    <rrn>916116130233</rrn>
    <card_acceptor>test.bg                12629314     BG</card_acceptor>
    <mcc>7995</mcc>
    <trace>1130233</trace>
    <type/>
    <card_sub_type/>
    <terminal_id>53b1f5eacc9edd</terminal_id>
    <unique_tran_id/>
  </payment_authorizations_response>
</payment_authorizations_responses>
```


The attributes in the root node payment authorization responses includes information about the pagination of the response.

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| @per_page | integer | number of entries per page |
| @page | integer | the current page |
| @total_count | integer | total number of all entries |
| @pages_count | integer | total number of pages |

# Processed Transactions

**PROCESSED TRANSACTION API**

The Processed Transaction API can be used to retrieve data about processed transactions.

#### SINGLE PROCESSED TRANSACTION

Single processed transaction retrieval allows to get a certain processed transaction by its ARN or by passing its unique ID.

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://staging.gate.e-comprocessing.net/processed_transactions \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <processed_transaction_request>
    <arn>74537604221431003881865</arn>
  </processed_transaction_request>'
```

OR

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://staging.gate.e-comprocessing.net/processed_transactions \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <processed_transaction_request>
    <unique_id>53b1f5eacc9e4d3a3afebb4e993fe962</unique_id>
  </processed_transaction_request>'
```

The URLs for the single processed transaction API are:

Production:

`https://gate.e-comprocessing.net/processed_transactions`

Staging (for integration):

`https://staging.gate.e-comprocessing.net/processed_transactions`

Successful Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<processed_transaction_response>
  <merchant_number>124000000006698</merchant_number>
  <batch_number>eMP</batch_number>
  <transaction_date>2019-09-01 16:43:02 UTC</transaction_date>
  <post_date>2019-09-01</post_date>
  <terminal_id>53b1f5eacc9eddda3afebb4e993fe962</terminal_id>
  <auth_code>09117B</auth_code>
  <currency>USD</currency>
  <amount>3690</amount>
  <merchant_transaction_reference>b76e9a54bdc99b3</merchant_transaction_reference>
  <arn>85301169244934771128812</arn>
  <card_brand>MC World Signia</card_brand>
  <card_number>420000...0000</card_number>
  <bin_country>124</bin_country>
  <service_type_desc>Credit Card</service_type_desc>
  <merchant_country>826</merchant_country>
  <area_of_event>Foreign - MASTER</area_of_event>
  <cross_rate>1</cross_rate>
  <card_scheme>Mastercard</card_scheme>
  <unique_id>b76e9a54bdc99b338068681727ed5e240000</unique_id>
  <type>purchase</type>
  <card_present>false</card_present>
  <deposit_slip_number>60506291293</deposit_slip_number>
  <batch_slip_number>60506282664</batch_slip_number>
  <fees>
    <fee>
      <type>Assessment fee</type>
      <amount>-0.74</amount>
      <currency>USD</currency>
      <charge_amount>-0.74</charge_amount>
      <charge_currency>USD</charge_currency>
    </fee>
  </fees>
</processed_transaction_response>
```


**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| merchant_number | string(20) | Merchant number |
| batch_number | string(23) | Batch number |
| transaction_date | string(255) | Transaction date in date and time format |
| post_date | string(255) | Posting date |
| terminal_id | string(10) | Terminal ID |
| arn | string(23) | Acquirer reference number |
| merchant_transaction_reference | string(23) | Merchant's transaction reference number |
| card_scheme | string(16) | Descriptive text for the card scheme |
| service_type_desc | string(25) | Indicates if transaction is a Debit or Credit transaction |
| card_brand | string(3) | Scheme card brand |
| card_number | string(13) | Masked card number |
| bin_country | string(3) | Issuing BIN ISO Country Code from Scheme BIN tables |
| merchant_country | string(3) | 3 digit ISO country code of the merchant country |
| area_of_event | string(19) | Area of event |
| currency | string(3) | Currency of transaction |
| cross_rate | float(11) | FX rate to convert from transaction account to merchant funding currency |
| auth_code | string(6) | Authorization code |
| unique_id | string(36) | Unique Transaction Identifier is generated at PoS before sent for authorisation or offline approval |
| card_present | boolean | Transaction is card present or card not present |
| deposit_slip_number | string(11) | Deposit Slip Number |
| batch_slip_number | string(11) | Batch Slip Number |
| type | string(65) | Transaction type for the related charge posted to account |
| **fees** | | |
| type | string(65) | Transaction type for the related charge posted to account |
| amount | float | Calculated charge amount in transaction in major currency unit, seeCurrency and Amount Handling for details. |

| Parameter | Type | Description |
|---|---|---|
| currency | string(3) | Currency of transaction |
| charge_amount | float | Calculated charge amount in charge currency in major currency unit, seeCurrency and Amount Handling for details. |
| charge_currency | string(3) | Currency of charge |

Error Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<processed_transaction_response>
  <status>error</status>
  <code>465</code>
  <message>Processed transaction not found!</message>
  <technical_message>Processed transaction by the given criteria cannot be found!</technical_message>
</processed_transaction_response>
```

In case no processed transaction is found for the given ARN or unique ID, a corresponding XML response is as follows:

#### BY DATE OR POST DATE RANGE

Date range based processed transaction retrieval allows you to fetch information for all processed transactions for a given merchant within a given date range. Date range searches for processed transactions either by their creation or posting date. The response is paginated, each request will return 100 entries max.

The URLs for date range processed transaction retrieval are:

Production:

`https://gate.e-comprocessing.net/processed_transactions/by_date`

`https://gate.e-comprocessing.net/processed_transactions/by_post_date`

Staging (for integration):

`https://staging.gate.e-comprocessing.net/processed_transactions/by_date`

`https://staging.gate.e-comprocessing.net/processed_transactions/by_post_date`

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://staging.gate.e-comprocessing.net/processed_transactions/by_post_date \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <processed_transaction_request>
    <start_date>2014-01-01</start_date>
    <end_date>2014-01-31</end_date>
    <batch_number>2065063</batch_number>
    <batch_slip_number>366236636</batch_slip_number>
    <deposit_slip_number>224234433</deposit_slip_number>
    <externally_processed>external</externally_processed>
    <processing_type>all</processing_type>
    <page>1</page>
  </processed_transaction_request>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| start_date | required | yyyy-mm-dd | start of the requested date range |
| end_date | optional | yyyy-mm-dd | end of the requested date range |
| page | optional | integer | the page within the paginated result, defaults to 1 |

| Parameter | Required | Format | Description |
|---|---|---|---|
| per_page | optional | integer | Number of entities on page, defaults to 100 |
| batch_number | optional | string(255) | Batch number of processed transactions (only for `by_post_date` API call) |
| batch_slip_number | optional | string(255) | Batch slip number of processed transactions (only for `by_post_date` API call) |
| deposit_slip_number | optional | string(255) | Deposit slip number of processed transactions (only for `by_post_date` API call) |
| externally_processed | optional | string(255) | Filters transactions by being externally processed or being native to Genesis. Possible values include 'genesis', 'external', and 'all'. If flag not supplied, it defaults to 'genesis' |
| processing_type | optional | string(255) | Filters transactions by being card present or card not present. Possible values include 'card_present', 'card_not_present', and 'all'. If flag not supplied, it defaults to 'all'. |

`required* = conditionally required`

Successful Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```


```xml
<?xml version="1.0" encoding="UTF-8"?>
<processed_transaction_responses per_page="100" page="1" total_count="2" pages_count="1">
  <processed_transaction_response>
    <merchant_number>1240000000006698</merchant_number>
    <batch_number>eMP</batch_number>
    <transaction_date>2019-09-01 16:43:02 UTC</transaction_date>
    <post_date>2019-09-01</post_date>
    <terminal_id>53b1f5eacc9eddda3afebb4e993fe962</terminal_id>
    <auth_code>09117B</auth_code>
    <currency>USD</currency>
    <amount>3690</amount>
    <merchant_transaction_reference>b76e9a54bdc99b3</merchant_transaction_reference>
    <card_brand>MC World Signia</card_brand>
    <card_number>420000...0000</card_number>
    <bin_country>124</bin_country>
    <service_type_desc>Credit Card</service_type_desc>
    <merchant_country>826</merchant_country>
    <area_of_event>Foreign - MASTER</area_of_event>
    <cross_rate>1</cross_rate>
    <card_scheme>Mastercard</card_scheme>
    <unique_id>b76e9a54bdc99b338068681727ed5e240000</unique_id>
    <type>purchase</type>
    <card_present>false</card_present>
    <deposit_slip_number>60506291293</deposit_slip_number>
    <batch_slip_number>60506282664</batch_slip_number>
    <arn>85301169244934771128812</arn>
  </processed_transaction_response>
  <processed_transaction_response>
    <merchant_number>1240000000006698</merchant_number>
    <batch_number>eMP</batch_number>
    <transaction_date>2019-09-01 16:43:02 UTC</transaction_date>
    <post_date>2019-09-01</post_date>
    <terminal_id>53b1f5eacc9eddda3afebb4e993fe962</terminal_id>
    <auth_code>09117B</auth_code>
    <currency>USD</currency>
    <amount>3690</amount>
    <merchant_transaction_reference>b76e9a54bdc99b3</merchant_transaction_reference>
    <card_brand>MC World Signia</card_brand>
    <card_number>420000...0000</card_number>
    <bin_country>124</bin_country>
    <service_type_desc>Credit Card</service_type_desc>
    <merchant_country>826</merchant_country>
    <area_of_event>Foreign - MASTER</area_of_event>
    <cross_rate>1</cross_rate>
    <card_scheme>Mastercard</card_scheme>
    <unique_id>b76e9a54bdc99b338068681727ed5e240000</unique_id>
    <type>purchase</type>
    <card_present>false</card_present>
    <deposit_slip_number>60506291293</deposit_slip_number>
    <batch_slip_number>60506282664</batch_slip_number>
    <arn>85301169244934771128812</arn>
    <fees>
      <fee>
        <type>Assessment fee</type>
        <amount>-0.74</amount>
        <currency>USD</currency>
        <charge_amount>-0.74</charge_amount>
        <charge_currency>USD</charge_currency>
      </fee>
    </fees>
  </processed_transaction_response>
</processed_transaction_responses>
```


The attributes in the root node processed transaction responses includes information about the pagination of the response.

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| @per_page | integer | number of entries per page |
| @page | integer | the current page |
| @total_count | integer | total number of all entries |
| @pages_count | integer | total number of pages |

# Processed Batches

#### PROCESSED BATCHES API

The Processed Batches API can be used to retrieve data about processed batches.

#### BY DATE OR POST DATE RANGE

Date range based processed batch retrieval allows you to fetch information for all processed batches for a given merchant within a given date range. Date range searches for processed batchess by their posting date. The response is paginated, each request will return 100 entries max.

The URLs for date range processed batch retrieval are:

Production:

`https://gate.e-comprocessing.net/processed_batches/by_post_date`

Staging (for integration):

`https://staging.gate.e-comprocessing.net/processed_batches/by_post_date`

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://staging.gate.e-comprocessing.net/processed_batches/by_post_date \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <processed_batch_request>
   <start_date>2014-01-01</start_date>
   <end_date>2014-01-31</end_date>
   <batch_slip_number>3354345543</batch_slip_number>
   <page>1</page>
 </processed_batch_request>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| start_date | required | yyyy-mm-dd | start of the requested date range |
| end_date | optional | yyyy-mm-dd | end of the requested date range |
| batch_slip_number | optional | string(255) | Batch slip number of processed transactions |
| page | optional | integer | the page within the paginated result, defaults to 1 |
| per_page | optional | integer | Number of entities on page, defaults to 100 |

`required* = conditionally required`

Successful Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<processed_batch_responses per_page="100" page="1" total_count="2" pages_count="1">
  <processed_batch_response>
    <merchant_number>105000000002227</merchant_number>
    <batch_number>eMP</batch_number>
    <batch_slip_number>94280612195</batch_slip_number>
    <batch_date>2019-10-24</batch_date>
    <post_date>2019-10-24</post_date>
    <terminal_id>53b1f5eacc9eddda3afebb4e993fe962</terminal_id>
    <transaction_type>Misc. DR transaction</transaction_type>
    <number_of_transactions>3</number_of_transactions>
    <currency>EUR</currency>
    <amount>370</amount>
  </processed_batch_response>
</processed_batch_responses>
```

The attributes in the root node processed batch responses includes information about the pagination of the response.

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| per_page | integer | number of entries per page |
| page | integer | the current page |
| total_count | integer | total number of all entries |
| pages_count | integer | total number of pages |
| batch_number | string(23) | Batch number. |
| batch_slip_number | string(11) | Batch slip number of the transaction. |
| batch_date | yyyy-mm-dd | Batch date. |
| post_date | yyyy-mm-dd | Posting date. |
| terminal_id | string(10) | Terminal ID. |
| transaction_type | string(28) | The type of the transaction. |
| number_of_transactions | integer | The number of the transactions. |
| currency | string(3) | The currency of the transactions. |
| amount | float | The amount of the entry in major currency unit, see Currency and Amount Handling for details. |

# Chargebacks

Chargebacks are a special type of transactions as they cannot be triggered by the merchant. Charge- backs occur if a customer disputes to an item of his credit card bill at his issuing bank and the bank requests a chargeback. In this case, the amount is automatically refunded to the customers cc account and deducted from your merchant account.

Customers who initiate chargebacks will automatically be blocked for future transactions. For details, please contact our Risk team.

You can also see a chargeback overview in the merchant console under the Risk Management menu.

#### CHARGEBACK REVERSALS

The reversals could be split into two types. These are the chargeback reversals, which appear when the a chargeback dispute is cancelled (withdrawn) by the consumer/issuer and the representments, which appear when the merchant or the acquirer disputes an already received chargeback. Both of these chargeback event types are handled properly and integrated into the whole process of chargeback dispute procedure.

#### CHARGEBACK NOTIFICATIONS

You now have the option to receive API and/or email notifications for each chargeback event that occurs - e.g. for first chargebacks, second chargebacks, and representments. Enable this feature by emailing the IT Support team at tech-support@e-comprocessing.com with the chargeback notification URL if needed.

The email notifications are sent to the merchant user with role 'admin' which is configured for managing the merchant entity on the gateway platform. The API notifications are equal to Notification for asynchronous payments, please refer to the section Notification for asynchronous payments to understand how notifications work.

Chargeback Notification Example

```
?transaction_id=343d9040a671c45832ee5381860e2996
&terminal_token=f4266042a6131b666060beb75691341d78ee5b4f
&unique_id=57ff74d1ca8727f59f243de6d01ff027
&transaction_type=sale
&status=chargebacked
&signature=ab4348afa9830834df90069646e4ce66c39a5358
&amount=100
&event=chargeback
```

#### CHARGEBACK API

The Chargeback API can be used to retrieve data about chargebacks.

#### SINGLE CHARGEBACK

Single chargeback retrieval allows to get a certain chargeback by its ARN or by passing the unique ID of the original transaction.

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://staging.gate.e-comprocessing.net/chargebacks \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <chargeback_request>
    <arn>74537604221431003881865</arn>
  </chargeback_request>'
```

OR

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://staging.gate.e-comprocessing.net/chargebacks \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <chargeback_request>
    <original_transaction_unique_id>53b1f5eacc9e4d3a3afebb4e993fe962</original_transaction_unique_id>
  </chargeback_request>'
```

#### LIST OF CHARGEBACKS

Retrieve a list of chargebacks by ARN or by passing the unique ID of the original transaction.

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://staging.gate.e-comprocessing.net/chargebacks \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <chargeback_request>
    <arn>74537604221431003881865</arn>
    <mode>list</mode>
  </chargeback_request>'
```

OR

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://staging.gate.e-comprocessing.net/chargebacks \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <chargeback_request>
    <original_transaction_unique_id>53b1f5eacc9e4d3a3afebb4e993fe962</original_transaction_unique_id>
    <mode>list</mode>
  </chargeback_request>'
```

The URLs for single and list of chargebacks API are:

Production:

`https://gate.e-comprocessing.net/chargebacks`

Staging (for integration):

`https://staging.gate.e-comprocessing.net/chargebacks`

Successful Response

```
stdClass Object
(
    [type] => 1st Chargeback
    [post_date] => 2014-01-24
    [reason_code] => 4855
    [merchant_number] => 443344323459841
    [reason_description] => Non-receipt of merchandise
    [authorization_code] => 811714
    [batch_number] => 2093064
    [cnn] => 9002578764
    [merchant_transaction_reference] => b76e9a54bdc99b3
    [capture_method] => SET/3D-SET authenticated
    [amount] => -14625
    [currency] => USD
    [chargeback_amount] => 300.0
    [chargeback_currency] => EUR
    [original_transaction_amount] => 148.0
    [original_transaction_currency] => EUR
    [merchant_settlement_amount] => 148.0
    [merchant_settlement_currency] => EUR
    [network_settlement_amount] => 148.0
    [network_settlement_currency] => EUR
    [merchant_dba_name] => hyperstech.com
    [original_type] => Purchase
    [original_post_date] => 2019-06-28
    [original_transaction_date] => 2019-06-28
    [original_slip] => 92572791484
    [item_slip_number] => 93778283100
    [card_number] => 554960******5069
    [card_brand] => master
    [customer_email] => jonh_doe@example.com
    [customer_phone] => 3598851248512
    [transaction_type] => sale3d
    [original_transaction_unique_id] => f9634ec5e7dbe6ca3871974accb875cd
    [arn] => 74537604221431003881865
)
```

```
<payment_response content=[
    <type content=[1st Chargeback]>
    <post_date content=[2014-01-24]>
    <reason_code content=[4855]>
    <merchant_number content=[443344323459841]>
    <reason_description content=[Non-receipt of merchandise]>
    <authorization_code content=[811714]>
    <batch_number content=[2093064]>
    <cnn content=[9002578764]>
    <merchant_transaction_reference content=[b76e9a54bdc99b3]>
    <capture_method content=[SET/3D-SET authenticated]>
    <amount content=[-14625]>
    <currency content=[USD]>
    <chargeback_amount content=[300.0]>
    <chargeback_currency content=[EUR]>
    <original_transaction_amount content=[148.0]>
    <original_transaction_currency content=[EUR]>
    <merchant_settlement_amount content=[148.0]>
    <merchant_settlement_currency content=[EUR]>
    <network_settlement_amount content=[148.0]>
    <network_settlement_currency content=[EUR]>
    <merchant_dba_name content=[hyperstech.com]>
    <original_type content=[Purchase]>
    <original_post_date content=[2019-06-28]>
    <original_transaction_date content=[2019-06-28]>
    <original_slip content=[92572791484]>
    <item_slip_number content=[93778283100]>
    <card_number content=[554960******5069]>
    <card_brand content=[master]>
    <customer_email content=[jonh_doe@example.com]>
    <customer_phone content=[3598851248512]>
    <transaction_type content=[sale3d]>
    <original_transaction_unique_id content=[f9634ec5e7dbe6ca3871974accb875cd]>
    <arn content=[74537604221431003881865]>
]>
```



```
{
    type: "1st Chargeback",
    post_date: "2014-01-24",
    reason_code: "4855",
    merchant_number: "443344323459841",
    reason_description: "Non-receipt of merchandise",
    authorization_code: "811714",
    batch_number: "2093064",
    cnn: "9002578764",
    merchant_transaction_reference: "b76e9a54bdc99b3",
    capture_method: "SET/3D-SET authenticated",
    amount: "-14625",
    currency: "USD",
    chargeback_amount: "300.0",
    chargeback_currency: "EUR",
    original_transaction_amount: "148.0",
    original_transaction_currency: "EUR",
    merchant_settlement_amount: "148.0",
    merchant_settlement_currency: "EUR",
    network_settlement_amount: "148.0",
    network_settlement_currency: "EUR",
    merchant_dba_name: "hyperstech.com",
    original_type: "Purchase",
    original_post_date: "2019-06-28",
    original_transaction_date: "2019-06-28",
    original_slip: "92572791484",
    item_slip_number: "93778283100",
    card_number: "554960******5069",
    card_brand: "master",
    customer_email: "jonh_doe@example.com",
    customer_phone: "3598851248512",
    transaction_type: "sale3d",
    original_transaction_unique_id: "f9634ec5e7dbe6ca3871974accb875cd",
    arn: "74537604221431003881865",
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<chargeback_response>
  <type>1st Chargeback</type>
  <post_date>2014-01-24</post_date>
  <reason_code>4855</reason_code>
  <merchant_number>443344323459841</merchant_number>
  <reason_description>Non-receipt of merchandise</reason_description>
  <authorization_code>811714</authorization_code>
  <batch_number>2093064</batch_number>
  <cnn>9002578764</cnn>
  <merchant_transaction_reference>b76e9a54bdc99b3</merchant_transaction_reference>
  <capture_method>SET/3D-SET authenticated</capture_method>
  <amount>-14625</amount>
  <currency>USD</currency>
  <chargeback_amount>300.0</chargeback_amount>
  <chargeback_currency>EUR</chargeback_currency>
  <original_transaction_amount>148.0</original_transaction_amount>
  <original_transaction_currency>EUR</original_transaction_currency>
  <merchant_settlement_amount>148.0</merchant_settlement_amount>
  <merchant_settlement_currency>EUR</merchant_settlement_currency>
  <network_settlement_amount>148.0</network_settlement_amount>
  <network_settlement_currency>EUR</network_settlement_currency>
  <merchant_dba_name>hyperstech.com</merchant_dba_name>
  <original_type>Purchase</original_type>
  <original_post_date>2019-06-28</original_post_date>
  <original_transaction_date>2019-06-28</original_transaction_date>
  <original_slip>92572791484</original_slip>
  <item_slip_number>93778283100</item_slip_number>
  <card_number>554960******5069</card_number>
  <card_brand>master</card_brand>
  <customer_email>jonh_doe@example.com</customer_email>
  <customer_phone>3598851248512</customer_phone>
  <transaction_type>sale3d</transaction_type>
  <original_transaction_unique_id>f9634ec5e7dbe6ca3871974accb875cd</original_transaction_unique_id>
  <arn>74537604221431003881865</arn>
</chargeback_response>
```


**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| type | string(255) | The chargeback type. See chargeback types for details |
| post_date | string(255) | The date of the chargeback |
| reason_code | string(255) | Reason code of the chargeback |
| merchant_number | string(255) | Merchant number |
| reason_description | string(255) | Reason description of the chargeback |
| authorization_code | string(255) | Authorization code of the chargeback's transaction |
| batch_number | string(23) | The batch number is provided by the submitter of the original presentment |
| cnn | string(14) | Chargeback Control Number filled for chargebacks and representments; empty for transfer transactions. |
| merchant_transaction_reference | string(255) | Merchant's transaction reference number |
| capture_method | string(255) | Capture method |
| amount | integer | Amount in minor units in cardholder account currency. Amount can be negative for: 1st chargeback, 2nd chargeback, transfer reversal and positive for all other types, see Currency and Amount Handling for details. |
| currency | string(3) | Currency of the cardholder account. See ISO 4217 |
| chargeback_amount | float | The amount in the chargeback's transaction currency, see Currency and Amount Handling for details. |
| chargeback_currency | string(3) | The currency of the chargeback's transaction. See ISO 4217 |
| original_transaction_amount | float(18) | Amount of the original presentment in transaction currency, see Currency and Amount Handling for details. |
| original_transaction_currency | string(3) | Transaction currency of the original presentment |
| merchant_settlement_amount | float(18) | Amount settled with the merchant for the original presentment (that is, the amount posted to the merchant account), before the deduction of any charges, see Currency and Amount Handling for details. |
| merchant_settlement_currency | string(3) | Currency settled with the payment network for the presentment before the deduction of any charges. |
| network_settlement_amount | float(18) | Amount settled with the merchant for the original presentment (that is, the amount posted to the merchant account), before the deduction of any charges, see Currency and Amount Handling for details. |
| network_settlement_currency | string(3) | Currency settled with the payment network for the presentment before the deduction of any charges. |
| merchant_dba_name | string(25) | Merchant name in the transaction as cleared to the schemes (charge descriptor). |
| original_type | string(28) | Transaction type of the original presentment |
| original_post_date | date(8) | Original presentment posting date |
| original_transaction_date | date(8) | Transaction date of the original presentment |

| Parameter | Type | Description |
|---|---|---|
| original_slip | string(11) | OmniPay internal slip number of the original presentment |
| item_slip_number | string(11) | OmniPay internal slip number of the original presentment |
| card_number | string(255) | Card number used for the chargeback's transaction |
| card_brand | string(255) | Card brand of the card number |
| customer_email | string(255) | The email of the cardholder |
| customer_phone | integer | The phone of the cardholder |
| transaction_type | string(255) | The type of the chargeback's transaction |
| original_transaction_unique_id | string(255) | The unique id of the chargeback's transaction |
| arn | string(255) | ARN of the chargeback's transaction |

Error Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<chargeback_response>
  <status>error</status>
  <code>470</code>
  <message>Chargeback not found!</message>
  <technical_message>Chargeback by the given criteria cannot be found!</technical_message>
</chargeback_response>
```

In case no chargeback is found for the given ARN or unique ID, a corresponding XML response is as follows:

#### BY DATE RANGE

Date range based chargeback retrieval allows you to fetch information for all chargebacks for a given merchant within a given date range. Date range searches for chargebacks by their posting date. Search option is chargeback retrieval by their import (creation) date. The response is paginated, each request will return 100 entries max.

The URLs for date range chargeback retrieval are:

Production:

`https://gate.e-comprocessing.net/chargebacks/by_date`

Staging (for integration):

`https://staging.gate.e-comprocessing.net/chargebacks/by_date`

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://staging.gate.e-comprocessing.net/chargebacks/by_date \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <chargeback_request>
   <start_date>2014-01-01</start_date>
   <end_date>2014-01-31</end_date>
   <externally_processed>external</externally_processed>
   <processing_type>all</processing_type>
   <page>1</page>
 </chargeback_request>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| start_date | required | yyyy-mm-dd | start of the requested date range |

| Parameter | Required | Format | Description |
|---|---|---|---|
| end_date | optional | yyyy-mm-dd | end of the requested date range |
| import_date | optional | yyyy-mm-dd | date of import in our system. Spans from beggining until end of day. |
| page | optional | integer | the page within the paginated result, defaults to 1 |
| per_page | optional | integer | Number of entities on page, defaults to 100 |
| externally_processed | optional | string(255) | Filters chargebacks by being externally processed or being native to Genesis. Possible values include 'genesis', 'external', and 'all'. If flag not supplied, it defaults to 'genesis' |
| processing_type | optional | string(255) | Filters chargebacks by being card present or card not present. Possible values include 'card_present', 'card_not_present', and 'all'. If flag not supplied, it defaults to 'all'. |

`required* = conditionally required`

Successful Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<chargeback_responses per_page="100" page="1" total_count="2" pages_count="1">
  <chargeback_response>
    <type>1st Chargeback</type>
    <merchant_number>443344323459841</merchant_number>
    <post_date>2014-01-24</post_date>
    <reason_code>4855</reason_code>
    <reason_description>Non-receipt of merchandise</reason_description>
    <authorization_code>811714</authorization_code>
    <batch_number>2093064</batch_number>
    <cnn>9002578764</cnn>
    <merchant_transaction_reference>b76e9a54bdc99b3</merchant_transaction_reference>
    <capture_method>SET/3D-SET authenticated</capture_method>
    <amount>-14625</amount>
    <currency>USD</currency>
    <chargeback_amount>300.0</chargeback_amount>
    <chargeback_currency>EUR</chargeback_currency>
    <original_transaction_amount>148.0</original_transaction_amount>
    <original_transaction_currency>EUR</original_transaction_currency>
    <merchant_settlement_amount>148.0</merchant_settlement_amount>
    <merchant_settlement_currency>EUR</merchant_settlement_currency>
    <network_settlement_amount>148.0</network_settlement_amount>
    <network_settlement_currency>EUR</network_settlement_currency>
    <merchant_dba_name>hyperstech.com</merchant_dba_name>
    <original_type>Purchase</original_type>
    <original_post_date>2019-06-28</original_post_date>
    <original_transaction_date>2019-06-28</original_transaction_date>
    <original_slip>92572791484</original_slip>
    <item_slip_number>93778283100</item_slip_number>
    <card_number>554960******5069</card_number>
    <card_brand>master</card_brand>
    <customer_email>jonh_doe@example.com</customer_email>
    <customer_phone>3598851248512</customer_phone>
    <transaction_type>sale3d</transaction_type>
    <original_transaction_unique_id>f9634ec5e7dbe6ca3871974accb875cd</original_transaction_unique_id>
    <arn>74537604221431003881865</arn>
  </chargeback_response>
  <chargeback_response>
    <type>2nd Chargeback</type>
    <merchant_number>443344323459841</merchant_number>
    <post_date>2014-01-27</post_date>
    <reason_code>4855</reason_code>
    <reason_description>Non-receipt of merchandise</reason_description>
    <authorization_code>811714</authorization_code>
    <batch_number>2093064</batch_number>
    <cnn>9002578764</cnn>
    <merchant_transaction_reference>b76e9a54bdc99b3</merchant_transaction_reference>
    <capture_method>SET/3D-SET authenticated</capture_method>
    <amount>3456</amount>
    <currency>USD</currency>
    <chargeback_amount>300.0</chargeback_amount>
    <chargeback_currency>EUR</chargeback_currency>
    <original_transaction_amount>148.0</original_transaction_amount>
    <original_transaction_currency>EUR</original_transaction_currency>
    <merchant_settlement_amount>148.0</merchant_settlement_amount>
    <merchant_settlement_currency>EUR</merchant_settlement_currency>
    <network_settlement_amount>148.0</network_settlement_amount>
    <network_settlement_currency>EUR</network_settlement_currency>
    <merchant_dba_name>hyperstech.com</merchant_dba_name>
    <original_type>Purchase</original_type>
    <original_post_date>2019-06-28</original_post_date>
    <original_transaction_date>2019-06-28</original_transaction_date>
    <original_slip>92572791484</original_slip>
    <item_slip_number>93778283100</item_slip_number>
    <card_number>454360******5008</card_number>
    <card_brand>visa</card_brand>
    <customer_email>ivan@example.com</customer_email>
    <customer_phone>3598851248512</customer_phone>
    <transaction_type>sale</transaction_type>
    <original_transaction_unique_id>67fbebc172b743a164a3f3af3d010457</original_transaction_unique_id>
    <arn>74537604221431003881865</arn>
  </chargeback_response>
</chargeback_responses>
```


The attributes in the root node chargeback responses includes information about the pagination of the response.

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| @per_page | integer | number of entries per page |
| @page | integer | the current page |
| @total_count | integer | total number of all entries |
| @pages_count | integer | total number of pages |

**CHARGEBACK TYPES**

Chargebacks will have one of the following type:

| Status | Description |
|---|---|

| Status | Description |
|---|---|
| 1st Chargeback | The first stage of the dispute procedure raised by the issuer |
| 2nd Chargeback | Second stage of the dispute procedure raised by the issuer (MasterCard only) |
| 1st Chargeback Reversal | When the first chargeback is cancelled (withdrawn) by the issuer |
| 2nd Chargeback Reversal | When the second chargeback is cancelled (withdrawn) by the issuer (MasterCard only) |
| Transfer Reversal | An operation that sends the amount of the dispute to the merchant when the acquirer represents a chargeback |
| Re-presentment | Acquirer's defend of the issuer's (first) chargeback |
| Chargeback Transfer to Merchant Hold Acc | The money is taken from merchant's hold account in Omnipay and is sent to the issuer |
| Chargeback Transfer to Writeoff Account Acq | The acquirer is taking the loss for this chargeback |
| Chargeback Transfer to Paymnt Acct Retail | The money is taken from merchant's account in Omnipay and are sent to the issuer (as per the chargeback rules). |
| Chargeback Transfer to Writeoff SP | The Service Provider is taking the loss for the chargeback. |

# Retrieval Requests

Retrieval requests are a special type of transactions as they cannot be triggered by the merchant. Retrieval requests occur if the issuer requests additional documentation for a transaction. Retrieval requests do not have financial implication, but they indicate the issuer doubts a given transaction and can initiate a chargeback.

For details, please contact our Risk team.

You can see also see a retrieval request overview in the merchant console under the Risk Management menu.

## Retrieval Request notifications

You now have the option to receive API and/or email notifications for each retrieval request event. Enable this feature by emailing the IT Support team atech-support@e-comprocessing.com with the desired notification URL.

The email notifications are sent to the merchant user with role 'admin' which is configured for managing the merchant entity on the gateway platform. The API notifications are equal to Notification for asynchronous payments, please refer to the section Notification for asynchronous payments to understand how notifications work.

Retrieval Notification Example

```
?transaction_id=30450
&terminal_token=cd577214de104fa0dd9c28486b3c817fd08c89a6
&unique_id=5de39380bf7ac7e1fc31cbd7805dc0ec
&transaction_type=sale
&status=chargebacked
&signature=98676c91391094b823d1f521d06cc129195952f9
&amount=400
&currency=USD
&avs_response_code=5I
&avs_response_text=Response+provided+by+issuer+processor%3B+Address+information+not+verified
&reason_code=10
&reason_description=Dispute+Transaction
&post_date=2014-07-16
&arn=17b4646c093b025
&event=retrieval_request
```

## Retrieval Request API

The retrieval request API can be used to get info for retrieval requests.

## Single Retrieval Request

Single retrieval request retrieval allows to get a certain retrieval request by its ARN or by passing the unique ID of the original transaction.

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://staging.gate.e-comprocessing.net/retrieval_equests \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <retrieval_request_request>
    <arn>74537604221431003881865</arn>
  </retrieval_request_request>'
```

OR

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://staging.gate.e-comprocessing.net/retrieval_equests \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <retrieval_request_request>
    <original_transaction_unique_id>53b1f5eacc9e4d3a3afebb4e993fe962</original_transaction_unique_id>
  </retrieval_request_request>'
```

---

# List of Retrieval Request

Retrieve a list of retrieval requests by ARN or by passing the unique ID of the original transaction.

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://staging.gate.e-comprocessing.net/retrieval_equests \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <retrieval_request_request>
    <arn>74537604221431003881865</arn>
    <retrieval_request_request>all</retrieval_request_request>
  </retrieval_request_request>'
```

OR

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://staging.gate.e-comprocessing.net/retrieval_equests \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <retrieval_request_request>
    <original_transaction_unique_id>53b1f5eacc9e4d3a3afebb4e993fe962</original_transaction_unique_id>
    <mode>list</mode>
  </retrieval_request_request>'
```

The URLs for the single and list of retrieval requests API are:

Production:

`https://gate.e-comprocessing.net/retrieval_equests`

Staging (for integration):

`https://staging.gate.e-comprocessing.net/retrieval_requests`

Successful Response

```
stdClass Object
(
    [type] => Retrieval request
    [arn] => 74537604221431003881865
    [post_date] => 2014-01-24
    [reason_code] => 42
    [reason_description] => Cardholder request
    [authorization_code] => 811714
    [merchant_number] => 124000000006698
    [issuer_number] => 0000002884
    [item_slip_number] => 93778283100
    [original_type] => Purchase
    [original_slip] => 92572791484
    [original_batch_number] => 2093064
    [description] => Action Control ID=20140224019152
    [fulfillment_date] => 2019-06-28
    [original_post_date] => 2019-06-28
    [original_transaction_date] => 2019-06-28
    [original_transaction_amount] => 148.0
    [original_transaction_currency] => EUR
    [merchant_settlement_amount] => 148.0
    [merchant_settlement_currency] => EUR
    [network_settlement_amount] => 148.0
    [network_settlement_currency] => EUR
    [card_number] => 554960******5069
    [card_brand] => master
    [customer_email] => jonh_doe@example.com
    [customer_phone] => 3598851248512
    [transaction_type] => sale3d
    [original_transaction_unique_id] => f9634ec5e7dbe6ca3871974accb875cd
    [arn] => 74537604221431003881865
)
```

```
<payment_response content=[
    <type content=[Retrieval request]>
    <arn content=[74537604221431003881865]>
    <post_date content=[2014-01-24]>
    <reason_code content=[42]>
    <reason_description content=[Cardholder request]>
    <authorization_code content=[811714]>
    <merchant_number content=[124000000006698]>
    <issuer_number content=[0000002884]>
    <item_slip_number content=[93778283100]>
    <original_type content=[Purchase]>
    <original_slip content=[92572791484]>
    <original_batch_number content=[2093064]>
    <description content=[Action Control ID=20140224019152]>
    <fulfillment_date content=[2019-06-28]>
    <original_post_date content=[2019-06-28]>
    <original_transaction_date content=[2019-06-28]>
    <original_transaction_amount content=[148.0]>
    <original_transaction_currency content=[EUR]>
    <merchant_settlement_amount content=[148.0]>
    <merchant_settlement_currency content=[EUR]>
    <network_settlement_amount content=[148.0]>
    <network_settlement_currency content=[EUR]>
    <card_number content=[554960******5069]>
    <card_brand content=[master]>
    <customer_email content=[jonh_doe@example.com]>
    <customer_phone content=[3598851248512]>
    <transaction_type content=[sale3d]>
    <original_transaction_unique_id content=[f9634ec5e7dbe6ca3871974accb875cd]>
    <arn content=[74537604221431003881865]>
]>
```

```
{
    type: "Retrieval request",
    arn: "74537604221431003881865",
    post_date: "2014-01-24",
    reason_code: "42",
    reason_description: "Cardholder request",
    authorization_code: "811714",
    merchant_number: "124000000006698",
    issuer_number: "0000002884",
    item_slip_number: "93778283100",
    original_type: "Purchase",
    original_slip: "92572791484",
    original_batch_number: "2093064",
    description: "Action Control ID=20140224019152",
    fulfillment_date: "2019-06-28",
    original_post_date: "2019-06-28",
    original_transaction_date: "2019-06-28",
    original_transaction_amount: "148.0",
    original_transaction_currency: "EUR",
    merchant_settlement_amount: "148.0",
    merchant_settlement_currency: "EUR",
    network_settlement_amount: "148.0",
    network_settlement_currency: "EUR",
    card_number: "554960******5069",
    card_brand: "master",
    customer_email: "jonh_doe@example.com",
    customer_phone: "3598851248512",
    transaction_type: "sale3d",
    original_transaction_unique_id: "f9634ec5e7dbe6ca3871974accb875cd",
    arn: "74537604221431003881865",
}
```



```
<?xml version="1.0" encoding="UTF-8"?>
<retrieval_request_response>
  <type>Retrieval request</type>
  <arn>74537604221431003881865</arn>
  <post_date>2014-01-24</post_date>
  <reason_code>42</reason_code>
  <reason_description>Cardholder request</reason_description>
  <authorization_code>811714</authorization_code>
  <merchant_number>124000000006698</merchant_number>
  <issuer_number>0000002884</issuer_number>
  <item_slip_number>93778283100</item_slip_number>
  <original_type>Purchase</original_type>
  <original_slip>92572791484</original_slip>
  <original_batch_number>2093064</original_batch_number>
  <description>Action Control ID=20140224019152</description>
  <fulfillment_date>2019-06-28</fulfillment_date>
  <original_post_date>2019-06-28</original_post_date>
  <original_transaction_date>2019-06-28</original_transaction_date>
  <original_transaction_amount>148.0</original_transaction_amount>
  <original_transaction_currency>EUR</original_transaction_currency>
  <merchant_settlement_amount>148.0</merchant_settlement_amount>
  <merchant_settlement_currency>EUR</merchant_settlement_currency>
  <network_settlement_amount>148.0</network_settlement_amount>
  <network_settlement_currency>EUR</network_settlement_currency>
  <card_number>554960******5069</card_number>
  <card_brand>master</card_brand>
  <customer_email>jonh_doe@example.com</customer_email>
  <customer_phone>3598851248512</customer_phone>
  <transaction_type>sale3d</transaction_type>
  <original_transaction_unique_id>f9634ec5e7dbe6ca3871974accb875cd</original_transaction_unique_id>
</retrieval_request_response>
```


**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| type | string(255) | The retrieval request type. See retrieval request types for details |
| arn | string(255) | ARN of the retrieval request's transaction |
| post_date | string(255) | The date of the retrieval request |
| reason_code | string(255) | Reason code of the retrieval request |
| reason_description | string(255) | Reason description of the retrieval request |
| authorization_code | string(255) | Authorization code of the retrieval request's transaction |
| merchant_number | string(20) | Merchant number |
| issuer_number | string(14) | Issuer reference number for the retrieval request |
| item_slip_number | string(11) | OmniPay internal slip number of the original presentment |
| original_type | string(28) | Transaction type of the original presentment |
| original_slip | string(11) | OmniPay internal slip number of the original presentment |
| original_batch_number | string(23) | The batch number is provided by the submitter of the original presentment |

| Parameter | Type | Description |
|---|---|---|
| description | string(255) | Free-text note entered by the institution and associated with the fulfilment |
| fulfillment_date | date(8) | Date on which the retrieval request was fulfilled. Empty if not yet fulfilled |
| original_post_date | date(8) | Original presentment posting date |
| original_transaction_date | date(8) | Transaction date of the original presentment |
| original_transaction_amount | float | Amount of the original presentment in major currency unit, seeCurrency and Amount Handling for details. |
| original_transaction_currency | string(3) | Transaction currency of the original presentment |
| merchant_settlement_amount | float | Amount settled with the merchant for the original presentment (that is, the amount posted to the merchant account) in major currency unit, before the deduction of any charges, see Currency and Amount Handling for details. |
| merchant_settlement_currency | string(3) | Currency settled with the payment network for the presentment before the deduction of any charges. |
| network_settlement_amount | float | Amount settled with the merchant for the original presentment (that is, the amount posted to the merchant account) in major currency unit, before the deduction of any charges, see Currency and Amount Handling for details. |
| network_settlement_currency | string(3) | Currency settled with the payment network for the presentment before the deduction of any charges. |
| card_number | string(255) | Card number used for the retrieval request's transaction |
| card_brand | string(255) | Card brand of the card number |
| customer_email | string(255) | The email of the cardholder |
| customer_phone | integer | The phone of the cardholder |
| transaction_type | string(255) | The type of the retrieval request's transaction |
| original_transaction_unique_id | string(255) | The unique id of the retrieval request's transaction |
| arn | string(255) | ARN of the retrieval request's transaction |

Error Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<retrieval_request_response>
  <status>error</status>
  <code>480</code>
  <message>Retrieval request not found!</message>
  <technical_message>Retrieval request by the given criteria cannot be found!</technical_message>
</retrieval_request_response>
```

In case no retrieval request is found with the given ARN or unique ID, a corresponding XML error response is received.

---

## By date range

Date range based retrieval request retrieval allows you to fetch information for all retrieval requests for a given merchant within a given date range. Date range searches for retrieval requests by their posting date. The response is paginated, each request will return 100 entries max.

The URLs for date range retrieval request retrieval are:

Production:

`https://gate.e-comprocessing.net/retrieval_requests/by_date`

Staging (for integration):

`https://staging.gate.e-comprocessing.net/retrieval_requests/by_date`

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://staging.gate.e-comprocessing.net/retrieval_requests/by_date \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <retrieval_request_request>
    <start_date>2014-01-01</start_date>
    <end_date>2014-01-31</end_date>
    <page>1</page>
  </retrieval_request_request>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| start_date | required | yyyy-mm-dd | start of the requested date range |
| end_date | optional | yyyy-mm-dd | end of the requested date range |
| import_date | optional | yyyy-mm-dd | date of import in our system. Spans from beggining until end of day. |
| page | optional | integer | the page within the paginated result, defaults to 1 |
| per_page | optional | integer | Number of entities on page, defaults to 100 |
| externally_processed | optional | string(255) | Filters retrieval requests by being externally processed or being native to Genesis. Possible values include 'genesis', 'external', and 'all'. If flag not supplied, it defaults to 'genesis' |
| processing_type | optional | string(255) | Filters retrieval requests by being card present or card not present. Possible values include 'card_present', 'card_not_present', and 'all'. If flag not supplied, it defaults to 'all'. |

`required* = conditionally required`

Successful Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<retrieval_request_responses per_page="100" page="1" total_count="2" pages_count="1">
  <retrieval_request_response>
    <type>Retrieval request</type>
    <post_date>2014-01-24</post_date>
    <reason_code>42</reason_code>
    <reason_description>Cardholder request</reason_description>
    <authorization_code>811714</authorization_code>
    <merchant_number>1240000000006698</merchant_number>
    <issuer_number>0000002884</issuer_number>
    <item_slip_number>93778283100</item_slip_number>
    <original_type>Purchase</original_type>
    <original_slip>92572791484</original_slip>
    <original_batch_number>2093064</original_batch_number>
    <description>Action Control ID=20140224019152</description>
    <fulfillment_date>2019-06-28</fulfillment_date>
    <original_post_date>2019-06-28</original_post_date>
    <original_transaction_date>2019-06-28</original_transaction_date>
    <original_transaction_amount>148.0</original_transaction_amount>
    <original_transaction_currency>EUR</original_transaction_currency>
    <merchant_settlement_amount>148.0</merchant_settlement_amount>
    <merchant_settlement_currency>EUR</merchant_settlement_currency>
    <network_settlement_amount>148.0</network_settlement_amount>
    <network_settlement_currency>EUR</network_settlement_currency>
    <card_number>554960******5069</card_number>
    <card_brand>master</card_brand>
    <customer_email>jonh_doe@example.com</customer_email>
    <customer_phone>3598851248512</customer_phone>
    <transaction_type>sale3d</transaction_type>
    <original_transaction_unique_id>f9634ec5e7dbe6ca3871974accb875cd</original_transaction_unique_id>
    <arn>74537604221431003881865</arn>
  </retrieval_request_response>
  <retrieval_request_response>
    <type>Retrieval request</type>
    <post_date>2014-01-27</post_date>
    <reason_code>42</reason_code>
    <reason_description>Cardholder request</reason_description>
    <authorization_code>811714</authorization_code>
    <merchant_number>1240000000006698</merchant_number>
    <issuer_number>0000002884</issuer_number>
    <item_slip_number>93778283100</item_slip_number>
    <original_type>Purchase</original_type>
    <original_slip>92572791484</original_slip>
    <original_batch_number>2093064</original_batch_number>
    <description>Action Control ID=20140224019152</description>
    <fulfillment_date>2019-06-28</fulfillment_date>
    <original_post_date>2019-06-28</original_post_date>
    <original_transaction_date>2019-06-28</original_transaction_date>
    <original_transaction_amount>148.0</original_transaction_amount>
    <original_transaction_currency>EUR</original_transaction_currency>
    <merchant_settlement_amount>148.0</merchant_settlement_amount>
    <merchant_settlement_currency>EUR</merchant_settlement_currency>
    <network_settlement_amount>148.0</network_settlement_amount>
    <network_settlement_currency>EUR</network_settlement_currency>
    <card_number>454360******5008</card_number>
    <card_brand>visa</card_brand>
    <customer_email>ivan@example.net</customer_email>
    <customer_phone>3598851248512</customer_phone>
    <transaction_type>sale</transaction_type>
    <original_transaction_unique_id>67fbebc172b743a164a3f3af3d010457</original_transaction_unique_id>
    <arn>74537604221431003881865</arn>
  </retrieval_request_response>
</retrieval_request_responses>
```


The attributes in the root node **retrieval_request_responses** includes information about the pagination of the response.

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| @per_page | integer | number of entries per page |
| @page | integer | the current page |
| @total_count | integer | total number of all entries |
| @pages_count | integer | total number of pages |

# Funding Accounts

## Funding Accounts API

The Funding Accounts API can be used to retrieve data for funding accounts.

**BY POST DATE**

Fetch merchant related funding accounts by post date.

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://staging.gate.e-comprocessing.net/funding_accounts/by_post_date \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <funding_accounts_request>
   <start_date>2020-01-01</start_date>
   <end_date>2020-01-31</end_date>
   <externally_processed>external</externally_processed>
   <processing_type>all</processing_type>
   <page>1</page>
   <per_page>50</per_page>
 </funding_accounts_request>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| start_date | required | yyyy-mm-dd | Start of the requested post date range |
| end_date | optional | yyyy-mm-dd | End of the requested post date range |
| page | optional | integer | The page within the paginated result, defaults to 1 |
| per_page | optional | integer | Number of entities on page, defaults to 100 |
| externally_processed | optional | string(255) | Filters transactions by being externally processed or being native to Genesis. Possible values include 'genesis', 'external', and 'all'. If flag not supplied, it defaults to 'genesis' |
| processing_type | optional | string(255) | Filters transactions by being card present or card not present. Possible values include 'card_present', 'card_not_present', and 'all'. If flag not supplied, it defaults to 'all'. |

`required* = conditionally required`

The URLs for the funding accounts API are:

Production:

`https://gate.e-comprocessing.net/funding_accounts/by_post_date`

Staging (for integration):

`https://staging.gate.e-comprocessing.net/funding_accounts/by_post_date`

Successful Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<funding_accounts_responses>
  <funding_account_response>
    <merchant_number>505702571465826</merchant_number>
    <account_number>123445</account_number>
    <account_type>Merchant Account</account_type>
    <post_date>2020-01-02</post_date>
    <transaction_type>Funding</transaction_type>
    <slips>2</slips>
    <reference_number>60506291293</reference_number>
    <reversal_indicator>N</reversal_indicator>
    <currency>USD</currency>
    <amount>10.00</amount>
    <account_amount_gross>100.00</account_amount_gross>
    <account_total_charges>0.00</account_total_charges>
    <account_amount_net>100.00</account_amount_net>
    <card_brand>visa</card_brand>
    <batch_number>ERV</batch_number>
    <value_date>2019-12-12</value_date>
    <transaction_category>Charges &amp;amp; fees</transaction_category>
    <local_currency>USD</local_currency>
    <local_amount_gross>10.00</local_amount_gross>
    <account_currency>USD</account_currency>
  </funding_account_response>
</funding_accounts_responses>
```


**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| merchant_number | string(20) | Merchant number. |
| account_number | string(11) | OmniPay Unique Identifier for the Merchant account. |
| account_type | string(20) | The Type of the Account. |
| post_date | string(255) | The posting date on which the item in question was posted to the account. |
| transaction_type | string(65) | Transaction type description. For adjustment transactions this may include some explanatory text. |
| slips | string(10) | The number of internal slip numbers covered by the item. |
| reference_number | string(11) | This is a Slip Number in the OmniPay system for the transaction. |
| reversal_indicator | string(1) | Reversal Indicator Y if this is a reversal else N. |
| currency | string(3) | Swift code for the transaction currency of the item in question. |
| amount | float | Transaction amount of the item in major currency unit, see Currency and Amount Handling for details. |
| account_amount_gross | float(18,2) | The gross amount posted to the account, in the account currency rounded to 2 decimal places, see Currency and Amount Handling for details. |
| account_total_charges | float(18,2) | Applicable charges for the item, expressed in the account currency rounded to 2 decimals places, see Currency and Amount Handling for details. |
| account_amount_net | float(18,2) | The net amount in the account currency, after taking into account any charges rounded to 2 decimal places, see Currency and Amount Handling for details. |
| card_brand | string(16) | The card scheme associated to the transaction. Can be empty. |
| batch_number | string(23) | The 80byte batch number used to submit the transaction. This is only available for Transaction Types: -Merchant Purchase Deposits. -Merchant Debits. |
| arn | string(23) | Acquirer Reference Number. This is only available for the transaction Type: 'Chargeback'. |
| addendum_acquirer_reference | string(25) | ARN created by transferring a transaction or by Misc Batch Input. |
| fee_sequence | string(3) | Fee Sequence Code for a transaction. Usually provided for adjustment transaction in general it is empty. |
| fee_description | string(19) | Fee Sequence Description per Fee Sequence Code. |
| funding_date | string(255) | Associated Funding Date, only relevant for transactions whose category is 008 - Payments. |
| value_date | string(255) | Value Date. |
| transaction_category_code | string(3) | Transaction Category Code. |
| transaction_category | string(32) | Description for the transaction category code. |
| client_number | string(8) | This is the OmniPay Client Number and is the link to the MP feed. |
| payment_status | string(70) | The field will only be filled for payments if the institution uses the funding warehouse functionality. It will be empty for other transactions. |
| bank_reference | string(20) | This field is filled only for specific institutions on payment transactions. |
| account_type_code | string(3) | Account Type Code. |
| related_slip | string(11) | This field has details of a related transaction. Depending of the transaction type this could be a link to another transaction within the FA feed (Ref_no field) or a TR (Item_no field) or a PB (BTCH_SLIP field). |

| Parameter | Type | Description |
|---|---|---|
| transaction_type_code | string(3) | Transaction type code. |
| local_currency | string(3) | Swift code of the local currency. |
| local_amount_gross | string(21) | The gross amount posted to the account, in the local currency, seeCurrency and Amount Handling for details. |
| account_currency | string(3) | Account Currency. |

Invalid End Date

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://staging.gate.e-comprocessing.net/funding_accounts/by_post_date \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <funding_accounts_request>
   <start_date>2020-01-01</start_date>
   <end_date>13-2020</end_date>
 </funding_accounts_request>'
```

Error Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
 <?xml version="1.0" encoding="UTF-8"?>
 <funding_accounts_response>
   <status>error</status>
   <code>340</code>
   <message>Please check input data for errors!</message>
   <technical_message>input_error: 'end_date' has invalid format</technical_message>
 </funding_accounts_response>
```

# Fraud reports

SAFE/TC40 reports contain information for transactions reported as fraud to MasterCard or VISA.

You can see a SAFE/TC40 reports overview in the merchant console under the Risk management menu.

## SAFE/TC40 API

The SAFE/TC40 API can be used to retrieve data about SAFE/TC40 reports.

## Single SAFE/TC40 report

Single SAFE/TC40 retrieval allows to get a certain SAFE/TC40 by its ARN or by passing the unique ID of the original transaction.

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://staging.gate.e-comprocessing.net/fraud_reports \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <fraud_report_request>
    <arn>74537604221431003881865</arn>
  </fraud_report_request>'
```

OR

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://staging.gate.e-comprocessing.net/fraud_reports \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <fraud_report_request>
    <original_transaction_unique_id>53b1f5eacc9e4d3a3afebb4e993fe962</original_transaction_unique_id>
  </fraud_report_request>'
```

---

# List of SAFE/TC40 report

Retrieve a list of SAFE/TC40 by ARN or by passing the unique ID of the original transaction.

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://staging.gate.e-comprocessing.net/fraud_reports \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <fraud_report_request>
    <fraud_report_request>arn,</fraud_report_request>
    <mode>list</mode>
  </fraud_report_request>'
```

OR

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://staging.gate.e-comprocessing.net/fraud_reports \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <fraud_report_request>
    <original_transaction_unique_id>53b1f5eacc9e4d3a3afebb4e993fe962</original_transaction_unique_id>
    <mode>list</mode>
  </fraud_report_request>'
```

The URLs for the single and list of SAFE/TC40 API are:

Production:

`https://gate.e-comprocessing.net/fraud_reports`

Staging (for integration):

`https://staging.gate.e-comprocessing.net/fraud_reports`

Successful Response

```
stdClass Object
(
    [transaction_date] => 2014-01-24
    [fraud_type_code] => 6
    [amount] => 400
    [currency] => GBP
    [card_number] => 554960******5069
    [card_brand] => master
    [customer_email] => jonh_doe@example.com
    [customer_phone] => 3598851248512
    [transaction_type] => sale3d
    [original_merchant_transaction_id] => 123294sss
    [original_transaction_unique_id] => f9634ec5e7dbe6ca3871974accb875cd
    [arn] => 74537604221431003881865
)
```

```
<payment_response content=[
    <transaction_date content=[2014-01-24]>
    <fraud_type_code content=[6]>
    <amount content=[400]>
    <currency content=[GBP]>
    <card_number content=[554960******5069]>
    <card_brand content=[master]>
    <customer_email content=[jonh_doe@example.com]>
    <customer_phone content=[3598851248512]>
    <transaction_type content=[sale3d]>
    <original_merchant_transaction_id content=[123294sss]>
    <original_transaction_unique_id content=[f9634ec5e7dbe6ca3871974accb875cd]>
    <arn content=[74537604221431003881865]>
]>
```

```
{
    transaction_date: "2014-01-24",
    fraud_type_code: "6",
    amount: "400",
    currency: "GBP",
    card_number: "554960******5069",
    card_brand: "master",
    customer_email: "jonh_doe@example.com",
    customer_phone: "3598851248512",
    transaction_type: "sale3d",
    original_merchant_transaction_id: "123294sss",
    original_transaction_unique_id: "f9634ec5e7dbe6ca3871974accb875cd",
    arn: "74537604221431003881865",
}
```

```
  <?xml version="1.0" encoding="UTF-8"?>
  <fraud_report_response>
    <transaction_date>2014-01-24</transaction_date>
    <fraud_type_code>6</fraud_type_code>
    <amount>400</amount>
    <currency>GBP</currency>
    <card_number>554960******5069</card_number>
    <card_brand>master</card_brand>
    <customer_email>jonh_doe@example.com</customer_email>
    <customer_phone>3598851248512</customer_phone>
    <transaction_type>sale3d</transaction_type>
    <original_merchant_transaction_id>123294sss</original_merchant_transaction_id>
    <original_transaction_unique_id>f9634ec5e7dbe6ca3871974accb875cd</original_transaction_unique_id>
    <arn>74537604221431003881865</arn>
  </fraud_report_response>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|

| Parameter | Type | Description |
|---|---|---|
| transaction_date | yyyy-mm-dd | The date transaction is made |
| fraud_type_code | string(2) | 0 = Lost; 1 = Stolen; 2 = Card not received as issued (NRI); 3 = Fraudulent application; 4 = Issuer-reported counterfeit; 5 = Miscellaneous/Account takeover; 6 = Fraudulent use of account number; 7 = (U.S. only) used by ICS; 8 = (U.S. only) used by ICS; 9 = Acquirer-reported counterfeit |
| amount | integer | Amount in minor units in cardholder account currency. Amount can be negative for: 1st chargeback, 2nd chargeback, transfer reversal and positive for all other types. |
| currency | string(3) | Currency of the cardholder account. See ISO 4217 |
| card_number | string(255) | Card number used for the chargeback's transaction |
| card_brand | string(255) | Card brand of the card number |
| customer_email | string(255) | The email of the cardholder |
| customer_phone | integer | The phone of the cardholder |
| transaction_type | string(255) | The type of the chargeback's transaction |
| original_merchant_transaction_id | string(255) | Merchant reference id |
| original_transaction_unique_id | string(255) | The unique id of the chargeback's transaction |
| arn | string(255) | ARN of the chargeback's transaction |

Error Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<fraud_report_response>
  <status>error</status>
  <code>490</code>
  <message>Mastercard Fraud Report not found!</message>
  <technical_message>Mastercard fraud report by the given criteria cannot be found!</technical_message>
</fraud_report_response>
```

In case no SAFE/TC40 is found for the given ARN or unique ID, a corresponding XML response is as follows:

---

## By date range

Date range based SAFE/TC40 retrieval allows you to fetch information for all SAFE/TC40 reports for a given merchant within a given date range. Date range searches for SAFE/TC40 reports by their posting date. Search option is SAFE/TC40 retrieval by their import (creation) date. The response is paginated, each request will return 100 entries max.

The URLs for date range SAFE/TC40 retrieval are:

Production:

`https://gate.e-comprocessing.net/fraud_reports/by_date`

Staging (for integration):

`https://staging.gate.e-comprocessing.net/fraud_reports/by_date`

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://staging.gate.e-comprocessing.net/fraud_reports/by_date \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <fraud_reports>
   <start_date>2014-01-01</start_date>
   <end_date>2014-01-31</end_date>
   <page>1</page>
 </fraud_reports>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| start_date | required | yyyy-mm-dd | start of the requested date range |
| end_date | optional | yyyy-mm-dd | end of the requested date range |
| import_date | optional | yyyy-mm-dd | date of import in our system. Spans from beggining until end of day. |
| page | optional | integer | the page within the paginated result, defaults to 1 |
| per_page | optional | integer | Number of entities on page, defaults to 100 |

`required* = conditionally required`

Successful Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<fraud_report_responses per_page="100" page="1" total_count="2" pages_count="1">
  <fraud_report_response>
    <transaction_date>2014-01-24</transaction_date>
    <fraud_type_code>6</fraud_type_code>
    <amount>400</amount>
    <currency>GBP</currency>
    <card_number>554960******5069</card_number>
    <card_brand>master</card_brand>
    <customer_email>jonh_doe@example.com</customer_email>
    <customer_phone>3598851248512</customer_phone>
    <transaction_type>sale3d</transaction_type>
    <original_merchant_transaction_id>123294sss</original_merchant_transaction_id>
    <original_transaction_unique_id>f9634ec5e7dbe6ca3871974accb875cd</original_transaction_unique_id>
    <arn>74537604221431003881865</arn>
  </fraud_report_response>
  <fraud_report_response>
    <transaction_date>2014-01-27</transaction_date>
    <fraud_type_code>6</fraud_type_code>
    <amount>350</amount>
    <currency>GBP</currency>
    <card_number>454360******5008</card_number>
    <card_brand>visa</card_brand>
    <customer_email>peter@example.net</customer_email>
    <customer_phone>3598851248512</customer_phone>
    <transaction_type>sale</transaction_type>
    <original_merchant_transaction_id>123294sss</original_merchant_transaction_id>
    <original_transaction_unique_id>67fbebc172b743a164a3f3af3d010457</original_transaction_unique_id>
    <arn>74537604221431003881865</arn>
  </fraud_report_response>
</fraud_report_responses>
```

The attributes in the root node mastercard fraud report responses includes information about the pagination of the response.

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| @per_page | integer | number of entries per page |
| @page | integer | the current page |
| @total_count | integer | total number of all entries |
| @pages_count | integer | total number of pages |

# Blacklists

With the Blacklist API you can check if a certain credit card is blacklisted within the gateway. If a terminal token is not passed, the merchant and global PAN blacklists will be checked for the given card number. If a terminal token is passed, the terminal, its merchant, and the global PAN blacklists will be checked for blacklist matches.

The URLs for the Blacklist API are:

Production: `https://gate.e-comprocessing.net/blacklists`

Staging (for integration): `https://staging.gate.e-comprocessing.net/blacklists`

## Invoking a Request

A transaction is invoked via HTTPS POST, parameters are passed as XML with UTF-8 encoding.

```
<?xml version="1.0" encoding="UTF-8"?>
<blacklist_request>
  <card_number>4200000000000000</card_number>
  <terminal_token>abd30ed00f88f838c5d233cbb62b6da0b69267b4</terminal_token>
</blacklist_request>
```

| Parameter | Required | Format | Description |
|---|---|---|---|
| card_number | required | int(13..16) | the credit card number to be checked |
| terminal_token | optional | string(40) | the terminal token |

## Response

**Successful response:**

```
<?xml version="1.0" encoding="UTF-8"?>
<blacklist_response>
  <blacklisted>true</blacklisted>
</blacklist_response>
```

| Name | Type | Description |
|---|---|---|
| blacklisted | boolean | credit card number is blacklisted or not |

**Error response:**

```
<?xml version="1.0" encoding="UTF-8"?>
<blacklist_response>
  <code>350</code>
  <message>Invalid XML: No close tag for /blacklist_request</message>
</blacklist_response>
```

| Name | Type | Description |
|---|---|---|
| code | integer | error code of the error that occurred |
| message | string | info about the error |

# Asynchronous Transactions and Notifications

## Asynchronous Transactions

3-D secure transactions are always processed **asynchronously**, while the other transactions are processed **synchronously**.

This means that the final result of the transaction will not be available immediately and the status is pending async. Once the transaction has reached a final status, a Notification is sent to the merchant.

> Whenever the status is pending async the transaction gets processed asynchronously, and a [Notification](#notifications) is sent to the merchant once the transaction has reached a final status.

**Overview**

| Transaction type | async? |
|---|---|
| Authorize | never |
| Authorize3d | always |
| Sale | never |
| Sale3d | always |
| Capture | never |
| Refund | never |
| Async Refund | always |
| Void | never |
| InitRecurringSale | never |
| InitRecurringSale3D | always |
| RecurringSale | never |
| Credit | never |
| BitPay Sale | always |
| BitPay Refund | always |

# Notifications

For asynchronous payments a notification is always sent to the **notification_url** within the payment transaction.

The payment gateway can be configured to also send a notification after each synchronous payment transaction. The notification is sent to the **notification_url** which is configured per merchant.

Also see 3-D Secure Transactions and Notification for asynchronous payments.

The format of the notification in both cases is the same.

Notification Example

```
?transaction_id=82803B4C-70CC-43BD-8B21-FD0395285B40
&unique_id=44177a21403427eb96664a6d7e5d5d48
&transaction_type=sale3d
&terminal_token=394f2ebc3646d3c017fa1e1cbc4a1e20
&status=approved
&amount=500
&signature=088e16a1019277b15d58faf0541e11910eb756f6
&consumer_id=123456
&token=ee946db8-d7db-4bb7-b608-b65b153e127d
&eci=05
&avs_response_code=5I
&avs_response_text=Response+provided+by+issuer+processor%3B+Address+information+not+verified
```

**Parameters**

| Name | Type | Description |
|---|---|---|
| transaction_id | string | merchant generated transaction id |
| unique_id | string | unique id generated by Genesis |
| transaction_type | string | transaction type for the transaction eg: sale3d |
| terminal_token | string | the terminal token as used in the processing url |
| status | string | status of the payment transaction |
| amount | string | amount of the payment transaction. If the transaction is partially approved, this is the partially approved amount. Check Partial Approvals for details |
| partial_approval | string | If the transaction is partially approved, this is set to 'true'. Check Partial Approvals for details |
| signature | string | the signature of the notification, should be used to verify the notification was sent by Genesis |
| consumer_id | string(10) | Consumer unique reference. See Consumers |
| token | string(36) | Plain-text token value. See Tokenize |

| Name | Type | Description |
|---|---|---|
| eci | string | See Electronic Commerce Indicator as returned from the MPI for details |
| event | string | The event that caused the notification |
| rc_code | string | The reason code for the event |
| rc_description | string | The reason description for the event |
| avs_response_code | string | Generated by the card network on trying to match the billing ad- dress when performing the address verification. Optional, returned if config is enabled and acquirer supports it. Check AVS Status Codes for details. |
| avs_response_text | string | Gives the human response text for the AVS response code above. Optional, returned if config is enabled and acquirer supports it. Check AVS Status Codes for details. |
| reference_transaction_unique_id | string | The unique id generated by Genesis, identifies the reference transaction if present. |
| authorization code | string | A code returned by some acquirers to indicate that a card payment has been authorized. |
| retrieval_reference_number | string | A reference number used for tracking all messages related to a given cardholder transaction returned by some acquirers. |

Status will be either "declined", "approved" or "error", like shown in the states table.

The event parameter is added only for fraud transactions eg: chargeback, chargeback_reversal, representment, second_chargeback or retrieval_request.

The signature is a mean of security to ensure that the gate is really the sender of the notification. It is generated by concatenating the unique id of the transaction with your API password and generating a SHA1 Hash (Hex) of the string:

```
SHA1 Hash Hex of <unique_id><Your API password>
```

**Notification signature examples**

| unique_id | API password | signature |
|---|---|---|
| fc6c3c8c0219730c7a099eaa540f70dc | bogus | 08d01ae1ebdc22b6a1a764257819bb26e9e94e8d |
| 130319cfb3bf65ff3c4a4045487b173e | test123 | 1b34dabed996788efcc049567809484454ee8b17 |

You can use the signature to verify the integrity of the notification, ensuring that it was really sent by the gate.

> You must either use the signature to verify the notification's integrity or make a reconcile to check the final transaction status.

```
<?xml version="1.0" encoding="UTF-8"?>
<notification_echo>
    <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
</notification_echo>
```

When receiving the notification, you are required to render an xml page containing the transaction's unique id so that the gateway knows that you have accepted the notification. If the XML is not delivered, the notification is sent periodically until the XML is received.

# WPF

The WPF (Web Payment Form) is a customizable component of the Genesis payment gateway. It provides merchants with an intuitive user interface to easily process their customers' payments. Through a single point of integration, the merchant can offer his customers multiple payment methods instantly and since the WPF is hosted on the secure Genesis infrastructure, it is already PCI-DSS compliant.

## Workflow

In the example above the customer visits the merchant's website and does a checkout.

(1), the merchant initiates the payment on the Genesis payment gateway through a request to theWPF API, which carries the mandatory, initial payload (e.g. amount, currency, etc.). The response to this request (if successful) is a redirect URL, which the merchants hands over to the customer. Following this redirect URL

(2), the customer is then directed to the actual payment form (WPF), which gets served from the Genesis servers. Because the merchant has previously transmitted most of the relevant payment information, the form is prefilled with these values and the customer only needs to add personal data. The customer then selects one of the payment methods offered by the merchant and fills in his payment information

(3) (e.g. credit card data). Upon completion the customer is redirected back to the merchant

(4)[1]. After the payment has been processed and reached a final state the merchant is sent a notification

(5) to the notification url supplied in the initial create request (1). The merchant must either use the notification's signature to verify the payment's integrity or make a reconcile

(6) to check the final payment status. However, we urge all merchants to always do a reconcile.

`1 - Particularly to the return success url defined by the merchant in his initial request. If the customer has selected an asynchronous payment method, he is redirected to the MPI provider before this step`

> 3D secure WPF payments are always performed asynchronously. After submitting the web payment form, the customer is redirected to the MPI provider to enter his personal data. In the case when cardholder is not enrolled customer is redirected to the failure url.
>
> As with all other asynchronous payment transactions, the return-, success- and cancel-URLs are only meant to display a useful page/message to the customer. A redirect of the customer to one of these URLs never gives any form of indication of the payment's state. To find out whether the payment has gone through or not the merchant must always wait for the notification or (even better) do a reconcile.
>
> Please also note that there is no specific order in which notification and redirect will occur (that means that the notification may also arrive before the customer's redirect).

---

## WPF API

**URLS**

**Create:**

The URL for the WPF API create method is:

`https://wpf.e-comprocessing.net/<locale>/wpf`

For the test system the URL is:

`https://staging.wpf.e-comprocessing.net/<locale>/wpf`

Note that if you do not submit one of the available locales, defaults to 'en' (English).

Check the WPF Internationalization (i18n) for details.

**Reconcile:**

The URL for the WPF API reconcile method is:

`https://wpf.e-comprocessing.net/wpf/reconcile`

For the test system the URL is:

`https://staging.wpf.e-comprocessing.net/wpf/reconcile`

**CREATE**

Request

```
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('WPF\Create');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setDescription('You are about to buy 3 shoes at www.shoes.com!')
        ->setNotificationUrl('https://www.example.com/notification')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setReturnCancelUrl('http://www.example.com/cancel.html')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setConsumerId('123456')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')
        ->setRememberCard('true')
        ->setLifetime('60')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')


        // Risk Params
        ->setRiskUserId('123456')


        // Transaction Types
        ->addTransactionType('authorize')
        ->addTransactionType('sale')
        ->setPayLater('true')


        // Reminders
        ->addReminder('email', '40')
        ->addReminder('sms', '10');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.wpf.WPFCreateRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;
import java.net.MalformedURLException;
import java.net.URL;

public class GenesisExample {
  public static void main() throws MalformedURLException {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    WPFCreateRequest request = new WPFCreateRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setDescription("You are about to buy 3 shoes at www.shoes.com!");
    request.setNotificationUrl(new URL("https://www.example.com/notification"));
    request.setReturnSuccessUrl(new URL("http://www.example.com/success"));
    request.setReturnFailureUrl(new URL("http://www.example.com/failure"));
    request.setReturnCancelUrl(new URL("http://www.example.com/cancel.html"));
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setConsumerId("123456");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");
    request.setRememberCard("true");
    request.setLifetime(60);


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");


    // Risk Params
    request.setRiskUserId("123456");


    // Transaction Types
    request.addTransactionType("authorize").done();
    request.addTransactionType("sale").done();
    request.setPayLater("true");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.wpf_create(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "description": "You are about to buy 3 shoes at www.shoes.com!",
    "notification_url": "https://www.example.com/notification",
    "return_success_url": "http://www.example.com/success",
    "return_failure_url": "http://www.example.com/failure",
    "return_cancel_url": "http://www.example.com/cancel.html",
    "amount": "100",
    "currency": "USD",
    "consumer_id": "123456",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "remember_card": "true",
    "lifetime": 60,
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    },
    "risk_params": {
      "user_id": "123456"
    },
    "transaction_types": [
      "authorize",
      "sale"
    ],
    "business_attributes": {
      "name_of_the_supplier": "Best Furniture"
    },
    "pay_later": true,
    "reminder_language": "en",
    "reminders": [
      {
        "channel": "email",
        "after": 40
      },
      {
        "channel": "sms",
        "after": 10
      }
    ]
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://staging.wpf.e-comprocessing.net/wpf \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <wpf_payment>
   <transaction_id>43671</transaction_id>
   <usage>40208 concert tickets</usage>
   <description>You are about to buy 3 shoes at www.shoes.com!</description>
   <notification_url>https://www.example.com/notification</notification_url>
   <return_success_url>http://www.example.com/success</return_success_url>
   <return_failure_url>http://www.example.com/failure</return_failure_url>
   <return_cancel_url>http://www.example.com/cancel.html</return_cancel_url>
   <amount>100</amount>
   <currency>USD</currency>
   <consumer_id>123456</consumer_id>
   <customer_email>travis@example.com</customer_email>
   <customer_phone>+1987987987987</customer_phone>
   <remember_card>true</remember_card>
   <lifetime>60</lifetime>
   <billing_address>
     <first_name>Travis</first_name>
     <last_name>Pastrana</last_name>
     <address1>Muster Str. 12</address1>
     <zip_code>10178</zip_code>
     <city>Los Angeles</city>
     <state>CA</state>
     <country>US</country>
   </billing_address>
   <risk_params>
     <user_id>123456</user_id>
   </risk_params>
   <transaction_types>
     <transaction_type moto="true" name="authorize" fx_rate_id="123"/>
     <transaction_type crypto="true" name="sale" fx_rate_id="123"/>
   </transaction_types>
   <business_attributes>
     <name_of_the_supplier>Best Furniture</name_of_the_supplier>
   </business_attributes>
   <pay_later>true</pay_later>
   <reminder_language>en</reminder_language>
   <reminders>
     <reminder>
       <channel>email</channel>
       <after>40</after>
     </reminder>
     <reminder>
       <channel>sms</channel>
       <after>10</after>
     </reminder>
   </reminders>
 </wpf_payment>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| usage | optional | string(255) | Description of the transaction for later use. Note, it is required to be submitted for 3DSecure transactions in asynchronous workflow that are using the 3DSv2 authentication protocol. |
| amount | required* | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | required* | string(3) | Currency code in ISO 4217 |
| description | optional | string | a text describing the reason of the payment (e.g. "you're buying concert tickets") |
| consumer_id | required* | string(10) | See Consumers and Tokenization. Saved cards will be listed for user to select |
| customer_email | required* | e-mail address | Must contain valid e-mail of customer |
| customer_phone | required* | string(32) | Must contain valid phone number of customer |
| notification_url | required* | url | URL at merchant where gateway sends outcome of transaction. |
| return_success_url | required* | url | URL where customer is sent to after successful payment |
| return_failure_url | required* | url | URL where customer is sent to after unsuccessful payment |
| return_cancel_url | required* | string | URL where customer is sent to when the customer cancels the payment process within the WPF |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |

| Parameter | Required | Format | Description |
|---|---|---|---|
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required* | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |
| **transaction_types** | required | | The transaction types that the merchant is willing to accept payments for |
| transaction_type | required | string | One of the available WPF transaction types. Multiple transaction type elements supported. Attribute 'name' contains the transaction type name in question |
| **business_attributes** | required* | | Check business attributes section. |
| event_start_date | required* | dd-mm-yyyy | The date when event starts in format dd-mm-yyyy |
| event_end_date | required* | dd-mm-yyyy | The date when event ends in format dd-mm-yyyy |
| event_organizer_id | required* | string | |
| event_id | required* | string | |
| date_of_order | required* | dd-mm-yyyy | The date when order was placed in format dd-mm-yyyy |
| delivery_date | required* | dd-mm-yyyy | Date of the expected delivery in format dd-mm-yyyy |
| name_of_the_supplier | required* | string | |
| remember_card | optional | "true" | See Tokenize. Offer the user the option to save cardholder details for future use (tokenize). |
| lifetime | optional | integer | number of minutes determining how long the WPF will be valid. Will be set to 30 minutes by default. Valid value ranges between 1 minute and 31 days given in minutes |
| **risk_params** | optional | | list of risk params as described in the Advanced risk management with RiskParams section |
| user_id (example) | optional | string | the customer's ID within the merchant's system (example) |
| session_id (example) | optional | string | the customer's session ID within the mer- chant's system (example) |
| **dynamic_descriptor_params** | optional | | |
| merchant_name | optional | string(25) | Allows to dynamically override the charge descriptor |
| merchant_city | optional | string(13) | Allows to dynamically override the merchant phone number |
| sub_merchant_id | optional | string(15) | Allows to dynamically override the sub-merchant ID. |
| pay_later | optional | "true" | Signifies whether the 'Pay Later' feature would be enabled on the WPF |
| reminder_language | optional | string | It must be a valid language abbreviation from the availableWPF languages |
| **reminders** | optional | | Settings for reminders sending when using the 'Pay Later' feature. The number of the sent reminders would be exactly as sent or configured and delivery failures could be handled on demand. Also there will be no reminders sent if the WPF is already completed |
| **reminder** | optional | | Settings for a single reminder. Upto three reminders are allowed |
| channel | optional | string | Channel for sending WPF reminder. Valid values are 'email' and 'sms' |
| after | optional | integer | Number of minutes after WPF creation when the reminder should be sent. Valid value ranges between 1 minute and 31 days given in minutes |

| Parameter | Required | Format | Description |
|---|---|---|---|
| crypto | optional | "true" | Signifies whether a purchase of crypto-currency transaction is performed. Must be populated when purchasing crypto-currency with a VISA card and MCC is 6051. Must be populated when purchasing crypto-currency with a MASTER or INTL MAESTRO card and MCC is one of 6051, 6211. Contact tech-support@e-comprocessing.com for more details |
| gaming | optional | "true" | Signifies whether a gaming transaction is performed. Gaming transactions usually use MCC 7995. Contacttech-support@e-comprocessing.com for more details |
| moto | optional | "true" | Signifies whether a MOTO (mail order telephone order) transaction is performed. Contacttech-support@e-comprocessing.com for more details |
| fx_rate_id | optional | integer | See Get rates for FX Service. Offers the option to use a specific FX rate to convert the transaction processing amount. Used FX rate should have the same source currency as the processing currency. Contact tech-support@e-comprocessing.com for more details |

`required* = conditionally required`

The 'amount' and 'currency' parameters are **NOT** required when all of the submitted transaction type(s) have no financial impact on consumer's account (i.e. 'account_verification')

The required business attributes must be provided with the WPF request if you are submitting at least one transaction type that supports business attributes

Successful Response

```
stdClass Object
(
    [transaction_type] => wpf_create
    [status] => new
    [mode] => live
    [transaction_id] => 43671
    [consumer_id] => 123456
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [technical_message] => Transaction successful!
    [message] => Transaction successful!
    [redirect_url] => https://staging.wpf.e-comprocessing.net/en/payment/c7e32c1e9d1
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:25.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [amount] => 100
    [currency] => USD
)
```

```
<payment_response content=[
    <transaction_type content=[wpf_create]>
    <status content=[new]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <consumer_id content=[123456]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <technical_message content=[Transaction successful!]>
    <message content=[Transaction successful!]>
    <redirect_url content=[https://staging.wpf.e-comprocessing.net/en/payment/c7e32c1e9d1]>
    <timestamp content=[2020-07-15T12:08:25Z]>
    <amount content=[100]>
    <currency content=[USD]>
]>
```

```
{
    transaction_type: "wpf_create",
    status: "new",
    mode: "live",
    transaction_id: "43671",
    consumer_id: "123456",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    technical_message: "Transaction successful!",
    message: "Transaction successful!",
    redirect_url: "https://staging.wpf.e-comprocessing.net/en/payment/c7e32c1e9d1",
    timestamp: "2020-07-15T12:08:25Z",
    amount: "100",
    currency: "USD",
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<wpf_payment>
  <transaction_type>wpf_create</transaction_type>
  <status>new</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <consumer_id>123456</consumer_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <technical_message>Transaction successful!</technical_message>
  <message>Transaction successful!</message>
  <redirect_url>https://staging.wpf.e-comprocessing.net/en/payment/c7e32c1e9d1</redirect_url>
  <timestamp>2020-07-15T12:08:25Z</timestamp>
  <amount>100</amount>
  <currency>USD</currency>
</wpf_payment>
```


Successful With Optional Invalid Transactions For Amount Response


```
stdClass Object
(
    [consumer_id] => 123456
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:25.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [amount] => 100
    [currency] => USD
    [redirect_url] => https://staging.wpf.e-comprocessing.net/en/payment/c7e32c1e9d1
    [invalid_transactions_for_amount] => alipay
)
```

```
<payment_response content=[
    <consumer_id content=[123456]>
    <timestamp content=[2020-07-15T12:08:25Z]>
    <amount content=[100]>
    <currency content=[USD]>
    <redirect_url content=[https://staging.wpf.e-comprocessing.net/en/payment/c7e32c1e9d1]>
    <invalid_transactions_for_amount content=[alipay]>
]>
```

```
{
    consumer_id: "123456",
    timestamp: "2020-07-15T12:08:25Z",
    amount: "100",
    currency: "USD",
    redirect_url: "https://staging.wpf.e-comprocessing.net/en/payment/c7e32c1e9d1",
    invalid_transactions_for_amount: "alipay",
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<wpf_payment>
  <consumer_id>123456</consumer_id>
  <timestamp>2020-07-15T12:08:25Z</timestamp>
  <amount>100</amount>
  <currency>USD</currency>
  <redirect_url>https://staging.wpf.e-comprocessing.net/en/payment/c7e32c1e9d1</redirect_url>
  <invalid_transactions_for_amount>alipay</invalid_transactions_for_amount>
</wpf_payment>
```


**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| consumer_id | string(10) | Consumer unique reference. See Consumers |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |
| redirect_url | url | URL where user has to be redirected to complete payment process. Contains the locale in which the web payment form will be rendered by default. See WPF Internationalization (i18n) |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08- 30T17:46:11Z |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |

| Parameter | Type | Description |
|---|---|---|
| currency | string(255) | Currency code in ISO 4217 |
| invalid_transactions_for_amount | string* | list of comma separated transactions for which amount is not within the allowed limit *present only when at least for one transaction type an amount is invalid, check Currency and Amount Handling for details |

Error Response

```
stdClass Object
(
    [transaction_type] => wpf_create
    [status] => error
    [mode] => live
    [transaction_id] => 43671
    [unique_id] => 44177a21403427eb96664a6d7e5d5d48
    [code] => 330
    [technical_message] => Unknown system error. Please contact support.
    [message] => Transaction failed, please contact support!
    [timestamp] => DateTime Object
        (
            [date] => 2020-07-15 12:08:25.000000
            [timezone_type] => 2
            [timezone] => Z
        )
    [amount] => 100
    [currency] => USD
)
```

```
<payment_response content=[
    <transaction_type content=[wpf_create]>
    <status content=[error]>
    <mode content=[live]>
    <transaction_id content=[43671]>
    <unique_id content=[44177a21403427eb96664a6d7e5d5d48]>
    <code content=[330]>
    <technical_message content=[Unknown system error. Please contact support.]>
    <message content=[Transaction failed, please contact support!]>
    <timestamp content=[2020-07-15T12:08:25Z]>
    <amount content=[100]>
    <currency content=[USD]>
]>
```

```
{
    transaction_type: "wpf_create",
    status: "error",
    mode: "live",
    transaction_id: "43671",
    unique_id: "44177a21403427eb96664a6d7e5d5d48",
    code: "330",
    technical_message: "Unknown system error. Please contact support.",
    message: "Transaction failed, please contact support!",
    timestamp: "2020-07-15T12:08:25Z",
    amount: "100",
    currency: "USD",
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<wpf_payment>
  <transaction_type>wpf_create</transaction_type>
  <status>error</status>
  <mode>live</mode>
  <transaction_id>43671</transaction_id>
  <unique_id>44177a21403427eb96664a6d7e5d5d48</unique_id>
  <code>330</code>
  <technical_message>Unknown system error. Please contact support.</technical_message>
  <message>Transaction failed, please contact support!</message>
  <timestamp>2020-07-15T12:08:25Z</timestamp>
  <amount>100</amount>
  <currency>USD</currency>
</wpf_payment>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| transaction_type | string(255) | The transaction type |
| status | string(255) | Status of the transaction, see states |
| mode | string(255) | Mode of the transaction's terminal, can be "live" or "test" |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| code | integer | Error code according to Error code table |

| Parameter | Type | Description |
|---|---|---|
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |

**NOTIFICATION**

WPF Notifications are sent once the WPF payment has reached a final state and are transmitted via HTTP POST (application/x-www-form-urlencoded) with the following parameters:

> Apart from the workflow described above, the Genesis system will also use the notification url endpoint to send notifications to the merchant if a WPF payment is chargebacked.

```
signature=c5219b3d385e74496b2b48a5497b347e102849f10eacd25b062f823b
&payment_transaction_transaction_type=sale
&payment_transaction_terminal_token=e9fd7a957845450fb7ab9dccb498b6e1f6e1e3aa
&payment_transaction_unique_id=bad08183a9ec545daf0f24c48361aa10
&payment_transaction_amount=500
&wpf_transaction_id=mtid20110408144716113553696 2
&wpf_status=approved
&wpf_unique_id=26aa150ee68b1b2d6758a0e6c44fce4c
&consumer_id=123456
&payment_transaction_token=ee946db8-d7db-4bb7-b608-b65b153e127d
&notification_type=wpf
&eci=05
&payment_transaction_avs_response_code=5I
&payment_transaction_avs_response_text=Response+provided+by+issuer+processor%3B+Address+information+not+verified
```

| Name | Type | Description |
|---|---|---|
| signature | string | the signature of the notification, should be used to verify the the notification was sent by Genesis |
| payment_transaction_transaction_type | string | transaction type for the transaction eg: sale3d |
| payment_transaction_terminal_token | string | the terminal token as used in the processing url |
| payment_transaction_unique_id | string | unique id generated by Genesis |
| payment_transaction_amount | string | Amount of the payment transaction. If the transaction is partially approved, this is the partially approved amount. CheckPartial Approvals for details |
| payment_transaction_partial_approval | string | If the transaction is partially approved, this is set to 'true'. CheckPartial Approvals for details |
| wpf_transaction_id | string | merchant generated tranaction id |
| wpf_status | string | status of the payment transaction |
| wpf_unique_id | string | unique id generated by Genesis, required for WPF payment reconciliation |
| consumer_id | string(10) | Consumer unique reference. SeeConsumers |
| payment_transaction_token | string(36) | Plain-text token value. SeeTokenize |
| notification_type | string | constant value "wpf" |
| eci | string | See Electronic Commerce Indicator as returned from the MPI for details |
| payment_transaction_avs_response_code | string | Generated by the card network on trying to match the billing address when performing the address verification. Optional, returned if config is enabled and acquirer supports it. Check AVS Status Codes for details. |
| payment_transaction_avs_response_text | string | Gives the human response text for the AVS response code above. Optional, returned if config is enabled and acquirer supports it. Check AVS Status Codes for details. |
| authorization code | string | A code returned by some acquirers to indicate that a card payment has been authorized. |
| retrieval_reference_number | string | A reference number used for tracking all messages related to a given cardholder transaction returned by some acquirers. |

Status will be one of the following: approved, declined, error, timeout, pending, refunded, voided, chargebacked, chargeback reversed, represented and second chargebacked.

The signature is a mean of security to ensure that the gate is really the sender of the notification. It is generated by concatenating the unique id of the payment with your API password and generating a SHA-512 Hash (Hex) of the string:

`SHA-512 Hash Hex of <wpf_unique_id><Your Merchant API password>`

**Notification signature examples**

| unique id | API password | signature |
|---|---|---|
| 26aa150ee68b1b2d6758a0e6c44fce4c | 50fd87e65eb415f42fb5af4c9cf497662e00b785 | c5219b3d385e74496b2b48a549 |
| 3f760162ef57a829011e5e2379b3fa17 | 50fd87e65eb415f42fb5af4c9cf497662e00b785 | 14519d0db2f7f8f407efccc9b099 |

> You must either use the signature to verify the notification's integrity or make a reconcile to check the final transaction status.

```
<?xml version="1.0" encoding="UTF-8"?>
<notification_echo>
  <wpf_unique_id>3f760162ef57a829011e5e2379b3fa17</wpf_unique_id>
</notification_echo>
```

When receiving the notification, you are required to render an xml page containing the transaction's unique id so that the gateway knows that you have accepted the notification. If the XML is not delivered, the notification is sent periodically until the XML is received.

**RECONCILE**

Reconcile can be used to retrieve data about a payment. This can be useful if you want to retrieve information about a payment whose status is timeout, which returned an error or has changed eg. has beed chargebacked.

Reconcile requests are handled exactly like transaction requests via XML. To a large degree, the WPF Reconcile follows the notions of the standard processing Reconcile API.

```
<?xml version="1.0" encoding="UTF-8"?>
<wpf_reconcile>
  <unique_id>26aa150ee68b1b2d6758a0e6c44fce4c</unique_id>
</wpf_reconcile>
```

| Parameter | Required | Format | Description |
|---|---|---|---|
| unique_id | required | string(32) | unique id as returned by the create request |

Please note that the response may include multiple payment transaction records, since a WPF payment is a container class for multiple payment transactions. Card brand and card number will be available in response only for card transaction types.

Example Reconcile Response XML for successful payment that is partially approved (check Partial Approvals for details):

```
<?xml version="1.0" encoding="UTF-8"?>
<wpf_payment>
  <status>approved</status>
  <unique_id>26aa150ee68b1b2d6758a0e6c44fce4c</unique_id>
  <transaction_id>mtid201104081447161135536962</transaction_id>
  <consumer_id>123456</consumer_id>
  <timestamp>2011-04-08T14:46:27Z</timestamp>
  <amount>5000</amount>
  <currency>USD</currency>
  <usage>Shopify Electronic Transaction</usage>
  <description>You are about to buy shoes from Shopify</description>
  <card_brand>visa</card_brand>
  <card_number>420000...0000</card_number>
  <card_type>CREDIT</card_type>
  <card_subtype>CARD SUBTYPE</card_subtype>
  <card_issuing_bank>Issuing Bank</card_issuing_bank>
  <card_holder>John Doe</card_holder>
  <expiration_year>2020</expiration_year>
  <expiration_month>2</expiration_month>
  <payment_transaction>
    <status>approved</status>
    <authorization_code>345678</authorization_code>
    <response_code>00</response_code>
    <transaction_type>sale</transaction_type>
    <unique_id>bad08183a9ec545daf0f24c48361aa10</unique_id>
    <transaction_id>mtid201104081447161135536962</transaction_id>
    <arn>74537605259536043849425</arn>
    <terminal_token>e9fd7a957845450fb7ab9dccb498b6e1f6e1e3aa</terminal_token>
    <mode>test</mode>
    <timestamp>2011-04-08T14:46:40Z</timestamp>
    <descriptor>e-comprocessing.net/bogus +49123456789</descriptor>
    <bank_account_number>Bank Account Number</bank_account_number>
    <bank_identifier_code>Bank Identifier Code</bank_identifier_code>
    <amount>4900</amount>
    <partial_approval>true</partial_approval>
    <currency>USD</currency>
    <customer_email>john.doe@example.com</customer_email>
    <customer_phone>+11234567890</customer_phone>
    <technical_message>TESTMODE: No real money will be transferred!</technical_message>
    <message>TESTMODE: No real money will be transferred!</message>
    <billing_address>
      <first_name>John</first_name>
      <last_name>Doe</last_name>
      <address1>32, Doestreet</address1>
      <address2></address2>
      <zip_code>12345</zip_code>
      <city>New York</city>
      <state>NY</state>
      <country>US</country>
    </billing_address>
  </payment_transaction>
</wpf_payment>
```

Example Response XML for voided payment:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<wpf_payment>
  <status>voided</status>
  <unique_id>26aa150ee68b1b2d6758a0e6c44fce4c</unique_id>
  <transaction_id>mtid2011040814471611355369 62</transaction_id>
  <consumer_id>123456</consumer_id>
  <timestamp>2011-04-08T14:46:27Z</timestamp>
  <amount>5000</amount>
  <currency>USD</currency>
  <usage>Shopify Electronic Transaction</usage>
  <description>You are about to buy shoes from Shopify</description>
  <card_brand>visa</card_brand>
  <card_number>420000...0000</card_number>
  <card_type>CREDIT</card_type>
  <card_subtype>CARD SUBTYPE</card_subtype>
  <card_issuing_bank>Issuing Bank</card_issuing_bank>
  <card_holder>John Doe</card_holder>
  <expiration_year>2020</expiration_year>
  <expiration_month>2</expiration_month>
  <payment_transaction>
    <status>voided</status>
    <authorization_code>345678</authorization_code>
    <response_code>00</response_code>
    <transaction_type>sale</transaction_type>
    <unique_id>bad08183a9ec545daf0f24c48361aa10</unique_id>
    <transaction_id>mtid2011040814471611355369 62</transaction_id>
    <arn>74537605259536043849425</arn>
    <terminal_token>e9fd7a957845450fb7ab9dccb498b6e1f6e1e3aa</terminal_token>
    <mode>test</mode>
    <timestamp>2011-04-08T14:46:40Z</timestamp>
    <descriptor>e-comprocessing.net/bogus +49123456789</descriptor>
    <bank_account_number>Bank Account Number</bank_account_number>
    <bank_identifier_code>Bank Identifier Code</bank_identifier_code>
    <amount>5000</amount>
    <currency>USD</currency>
    <customer_email>john.doe@example.com</customer_email>
    <customer_phone>+11234567890</customer_phone>
    <billing_address>
     <first_name>John</first_name>
     <last_name>Doe</last_name>
     <address1>32, Doestreet</address1>
     <address2></address2>
     <zip_code>12345</zip_code>
     <city>New York</city>
     <state>NY</state>
     <country>US</country>
    </billing_address>
  </payment_transaction>
  <payment_transaction>
    <status>approved</status>
    <authorization_code>345678</authorization_code>
    <response_code>00</response_code>
    <transaction_type>void</transaction_type>
    <unique_id>bad08183a9ec545daf0f24c48361aa10</unique_id>
    <transaction_id>mtid2011040814471611355369 62</transaction_id>
    <terminal_token>e9fd7a957845450fb7ab9dccb498b6e1f6e1e3aa</terminal_token>
    <mode>test</mode>
    <timestamp>2011-04-08T14:46:40Z</timestamp>
    <descriptor>e-comprocessing.net/bogus +49123456789</descriptor>
    <amount>5000</amount>
    <currency>USD</currency>
  </payment_transaction>
</wpf_payment>
```


Example Error Response XML:


```xml
<?xml version="1.0" encoding="UTF-8"?>
<wpf_payment>
  <status>error</status>
  <code>100</code>
  <technical_message>Unknown system error. Please contact support.</technical_message>
  <message>Transaction failed, please contact support!</message>
</wpf_payment>
```


**RECONCILE BY DATE RANGE**

Reconcile by Date can be used to retrieve data about payments in date and time range. This can be useful if the merchant wants to retrieve information about payments belonging to the date and time range. For convenience, there are options to define records per page and the exact page of interest.

Reconcile by Date requests are handled exactly like transaction requests in the Processing API. To a large degree, the WPF Reconcile follows the notions of the standard Reconcile in Processing API.

The URL for date range reconcile is:

`https://staging.wpf.e-comprocessing.net/wpf/reconcile/by_date`

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://staging.wpf.e-comprocessing.net/wpf/reconcile/by_date \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <wpf_reconcile>
   <start_date>2020-04-17 15:08:25</start_date>
   <end_date>2020-07-16 15:08:25</end_date>
   <page>1</page>
   <records_per_page>100</records_per_page>
 </wpf_reconcile>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| start_date | required | yyyy-mm-dd hh:mm:ss | Start date and time of the reconcile range (time is optional) |
| end_date | optional | yyyy-mm-dd hh:mm:ss | End of date and time of the reconcile range (time is optional) |
| page | optional | string(11) | Number of the page |
| records_per_page | optional | 1 to 3 digits | Number ot records per page |

`required* = conditionally required`

Successful Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
 <?xml version="1.0" encoding="UTF-8"?>
 <wpf_payment>
   <status>approved</status>
   <unique_id>e27bd9472d40d3734a49e3e5c5ada52d</unique_id>
   <transaction_id>246d15a8644ff203fca6ae39fac8e999</transaction_id>
   <timestamp>2020-06-26</timestamp>
   <amount>500</amount>
   <currency>USD</currency>
   <authorization_code>345678</authorization_code>
   <retrieval_reference_number>016813015184</retrieval_reference_number>
 </wpf_payment>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| status | string(255) | Status of the transaction, see states |
| unique_id | string(32) | Unique id defined by gate (must later be used if capturing, voiding or refunding a transaction) |
| transaction_id | string(255) | Unique transaction id defined by merchant |
| timestamp | string(255) | Time when the transaction was processed in ISO 8601 Combined date and time e.g. 2007-08- 30T17:46:11Z |
| amount | integer | Amount of transaction in minor currency unit, see Currency and Amount Handling for details |
| currency | string(255) | Currency code in ISO 4217 |
| authorization_code | string(6) | Generated by the card network when an authorisation has occurred, used to identify that auth. Consists of 6 alphanumeric chars |
| retrieval_reference_number | string(255) | A reference number used for tracking all messages related to a given cardholder transaction returned by some acquirers. |

Error Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<wpf_payment>
  <status>error</status>
  <code>340</code>
  <technical_message>start_date has an invalid format</technical_message>
  <message>Please check input data for errors!</message>
</wpf_payment>
```

Error Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<wpf_payment>
  <status>error</status>
  <code>320</code>
  <technical_message>start_date is missing</technical_message>
  <message>Please check input data for errors!</message>
</wpf_payment>
```

Error Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<wpf_payment>
  <status>error</status>
  <code>340</code>
  <technical_message>per_page has an invalid format</technical_message>
  <message>Please check input data for errors!</message>
</wpf_payment>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| status | string(255) | Status of the transaction, see states |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |

**ERRORS**

WPF returns the same error codes/XML as the regular Server-to-Server API methods (e.g. sale). See the Error Section for more details.

---

# WPF transaction types

The web payment form supports the typical card-related transaction types such as authorizes, sales, and init recurring - with and without 3D (see below). If more than one card-based transaction type is passed, then if the cardholder decides to pay with a credit card, the first card-based transaction type with a valid configuration - terminals, MIDs, currencies, etc - is selected and used when cardholder decides to pay with a (credit) card.

Also Account Verification for cards is supported by the web payment form. This is a transaction type for card verifications without any financial impact on cardholder's account. If 'account_verification' is the only transaction type provided, the 'amount' and the 'currency' parameters are **NOT** mandatory. If it is combined with other transaction types, they are still mandatory and required.

In addition, the WPF API supports a host of alternative payment methods (APMs), bank transfer payments, wallets, and more. All of the alternative transaction types are offered to the cardholder for payment, so the cardholder can choose between credit cards and alternatives defined by the merchant. Only credit cards or only a single alternative method can also work, there is no need to always offer credit cards for example.

Merchant custom attributes for a number of the transaction types are required. The WPF API supports those merchant custom attributes - they are submitted as child elements to the transaction type they belong to(see WPF API example request above).

The web payment form has the concept of a default payment method. If a given transaction type has the 'default' attribute set to 'true', then this transaction type will be pre-selected as default when the cardholder is redirected to the web payment form. If more than one transaction type has the 'default' attribute, the first one with this attribute will be pre-selected. In the case there is

only one transaction type requested (with or without the 'default'='true'), this transaction type is automatically set as default by design.

Note that for each transaction type requested by a merchant in the WPF API, a valid configuration should exist - valid terminal, currency, MID, web payment form enabled for the merchant, etc. - otherwise a configuration error will be raised. If this happens, contact the IT support team to resolve/configure your desired transaction types correctly.

Example Request XML with custom attributes:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<wpf_payment>
  <transaction_id>wev238f328nc</transaction_id>
  <usage>Order ID 500, Shoes</usage>
  <description>You are about to buy 3 shoes at www.shoes.com!</description>
  <notification_url>https://example.com/notification</notification_url>
  <return_success_url>https://example.com/return_success</return_success_url>
  <return_failure_url>https://example.com/return_failure</return_failure_url>
  <return_cancel_url>https://example.com/return_cancel</return_cancel_url>
  <amount>5000</amount>
  <currency>USD</currency>
  <customer_email>john.doe@example.com</customer_email>
  <customer_phone>+11234567890</customer_phone>
  <card_holder>john doe</card_holder>
  <billing_address>
    <first_name>John</first_name>
    <last_name>Doe</last_name>
    <address1>23, Doestreet</address1>
    <zip_code>11923</zip_code>
    <city>New York City</city>
    <state>NY</state>
    <country>US</country>
  </billing_address>
  <transaction_types>
    <transaction_type name="sale">
      <bin>420000</bin>
      <tail>0000</tail>
      <expiration_date>2017-03</expiration_date>
      <fx_rate_id>123</fx_rate_id>
      <crypto>true</crypto>
    </transaction_type>
    <transaction_type name="sale3d" default="true">
      <tail>1111</tail>
      <expiration_date>2016-03</expiration_date>
      <fx_rate_id>123</fx_rate_id>
      <crypto>true</crypto>
    </transaction_type>
    <transaction_type name="cashu"/>
    <transaction_type name="ezeewallet">
      <source_wallet_id>john.doe@e-comprocessing.net</source_wallet_id>
    </transaction_type>
    <transaction_type name="ppro">
        <payment_method>mybank</payment_method>
    </transaction_type>
    <transaction_type name="insta_debit_payin">
        <customer_account_id>123456</customer_account_id>
    </transaction_type>
  </transaction_types>
  <risk_params>
    <user_id>123456</user_id>
  </risk_params>
</wpf_payment>
```

| Transaction Type | Custom Attribute | Required | Description |
|---|---|---|---|
| **authorize** | | | A standard authorization |
| | bin | no | Card's first 6 digits |
| | tail | no | Card's last 4 digits |
| | default | no | Configure as default or not |
| | expiration_date | no | Expiration month and year |
| | gaming | no | Signifies whether a gaming transaction is performed. Gaming transactions usually use MCC 7995. Contacttech-support@e-comprocessing.com for more details |
| | crypto | no | Signifies whether a purchase of crypto-currency transaction is performed. Must be populated when purchasing crypto-currency with a VISA card and MCC is 6051. Must be populated when purchasing crypto-currency with a MASTER or INTL MAESTRO card and MCC is one of 6051, 6211. Contact tech-support@e-comprocessing.com for more details |
| | moto | no | Signifies whether a MOTO (mail order telephone order) transaction is performed. Contacttech-support@e-comprocessing.com for more details |
| | fx_rate_id | no | See Get rates for FX Service. Offers the option to use a specific FX rate to convert the transaction processing amount. Used FX rate should have the same source currency as the processing currency. Contact tech-support@e-comprocessing.com for more details |
| **authorize3d** | | | A 3D-based authorization |
| | bin | no | Card's first 6 digits |
| | tail | no | Card's last 4 digits |
| | default | no | Configure as default or not |

| Transaction Type | Custom Attribute | Required | Description |
|---|---|---|---|
| | expiration_date | no | Expiration month and year |
| | gaming | no | Signifies whether a gaming transaction is performed. Gaming transactions usually use MCC 7995. Contacttech-support@e-comprocessing.com for more details |
| | crypto | no | Signifies whether a purchase of crypto-currency transaction is performed. Must be populated when purchasing crypto-currency with a VISA card and MCC is 6051. Must be populated when purchasing crypto-currency with a MASTER or INTL MAESTRO card and MCC is one of 6051, 6211. Contact tech-support@e-comprocessing.com for more details |
| | moto | no | Signifies whether a MOTO (mail order telephone order) transaction is performed. Contacttech-support@e-comprocessing.com for more details |
| | fx_rate_id | no | See Get rates for FX Service. Offers the option to use a specific FX rate to convert the transaction processing amount. Used FX rate should have the same source currency as the processing currency. Contact tech-support@e-comprocessing.com for more details |
| **sale** | | | A standard sale |
| | bin | no | Card's first 6 digits |
| | tail | no | Card's last 4 digits |
| | default | no | Configure as default or not |
| | expiration_date | no | Expiration month and year |
| | gaming | no | Signifies whether a gaming transaction is performed. Gaming transactions usually use MCC 7995. Contacttech-support@e-comprocessing.com for more details |
| | crypto | no | Signifies whether a purchase of crypto-currency transaction is performed. Must be populated when purchasing crypto-currency with a VISA card and MCC is 6051. Must be populated when purchasing crypto-currency with a MASTER or INTL MAESTRO card and MCC is one of 6051, 6211. Contact tech-support@e-comprocessing.com for more details |
| | moto | no | Signifies whether a MOTO (mail order telephone order) transaction is performed. Contacttech-support@e-comprocessing.com for more details |
| | fx_rate_id | no | See Get rates for FX Service. Offers the option to use a specific FX rate to convert the transaction processing amount. Used FX rate should have the same source currency as the processing currency. Contact tech-support@e-comprocessing.com for more details |
| **sale3d** | | | A 3D-based sale |
| | bin | no | Card's first 6 digits |
| | tail | no | Card's last 4 digits |
| | default | no | Configure as default or not |
| | expiration_date | no | Expiration month and year |
| | gaming | no | Signifies whether a gaming transaction is performed. Gaming transactions usually use MCC 7995. Contacttech-support@e-comprocessing.com for more details |
| | crypto | no | Signifies whether a purchase of crypto-currency transaction is performed. Must be populated when purchasing crypto-currency with a VISA card and MCC is 6051. Must be populated when purchasing crypto-currency with a MASTER or INTL MAESTRO card and MCC is one of 6051, 6211. Contact tech-support@e-comprocessing.com for more details |
| | moto | no | Signifies whether a MOTO (mail order telephone order) transaction is performed. Contacttech-support@e-comprocessing.com for more details |
| | fx_rate_id | no | See Get rates for FX Service. Offers the option to use a specific FX rate to convert the transaction processing amount. Used FX rate should have the same source currency as the processing currency. Contact tech-support@e-comprocessing.com for more details |
| **init_recurring_sale** | | | A standard init recurring |
| | bin | no | Card's first 6 digits |
| | tail | no | Card's last 4 digits |
| | default | no | Configure as default or not |
| | expiration_date | no | Expiration month and year |
| | fx_rate_id | no | See Get rates for FX Service. Offers the option to use a specific FX rate to convert the transaction processing amount. Used FX rate should have the same source currency as the processing currency. Contact tech-support@e-comprocessing.com for more details |
| **init_recurring_sale3d** | | | A 3D-based init recurring |
| | bin | no | Card's first 6 digits |
| | tail | no | Card's last 4 digits |
| | default | no | Configure as default or not |
| | expiration_date | no | Expiration month and year |

| Transaction Type | Custom Attribute | Required | Description |
|---|---|---|---|
| | fx_rate_id | no | See Get rates for FX Service. Offers the option to use a specific FX rate to convert the transaction processing amount. Used FX rate should have the same source currency as the processing currency. Contact tech-support@e-comprocessing.com for more details |
| **account_verification** | | | Card verification without any financial impact |
| | bin | no | Card's first 6 digits |
| | tail | no | Card's last 4 digits |
| | default | no | Configure as default or not |
| | expiration_date | no | Expiration month and year |
| | moto | no | Signifies whether a MOTO (mail order telephone order) transaction is performed. Contacttech-support@e-comprocessing.com for more details |
| **ezeewallet** | | | Wallet-based payment |
| | source_wallet_id | no | Email address of consumer who owns the source wallet |
| | default | no | Configure as default or not |
| **sofort** | | | Bank transfer payment, popular in Germany |
| | default | no | Configure as default or not |
| **cashu** | | | Voucher-based payment |
| | default | no | Configure as default or not |
| **paysafecard** | | | Voucher-based payment |
| | default | no | Configure as default or not |
| **ppro** | | | Supports payments with EPS, TeleIngreso, SafetyPay, TrustPay, Przelewy24, iDEAL, QIWI, GiroPay, Mr. Cash and MyBank |
| | default | no | Configure as default or not |
| **neteller** | | | Wallet-based payment |
| | default | no | Configure as default or not |
| **poli** | | | Payment by bank account for customers with an Australian or New Zealand bank account. |
| = default | no | Configure as default or not | |
| **p24** | | | Payment by bank account for customers with a Polish bank account. |
| | default | no | Configure as default or not |
| **idebit_payin** | | | Online Banking ePayment method |
| | default | no | Configure as default or not |
| | customer_account_id | yes | Identifier provided by the merchant that uniquely identifies the consumer in their system |
| **insta_debit_payin** | | | Online Banking ePayment method |
| | default | no | Configure as default or not |
| | customer_account_id | yes | Identifier provided by the merchant that uniquely identifies the consumer in their system |
| **paypal_express** | | | Express payment by PayPal balance. |
| | default | no | Configure as default or not |
| **webmoney** | | | Bank transfer payment, popular in Russian Federation. |
| **sdd_sale** | | | SEPA Direct Debit Sale |
| **sdd_init_recurring_sale** | | | SEPA Direct Debit init recurring |
| **trustly_sale** | | | Solution for Online Banking ePayments. |
| **trustly_withdrawal** | | | Allows you to pay with your online bank. |
| **wechat** | | | Online Banking ePayment method |
| | product_code | no | Product code |
| | product_num | no | Product number |

| Transaction Type | Custom Attribute | Required | Description |
|---|---|---|---|
| | product_desc | no | Product description |

## WPF Internationalization (i18n)

The web payment form is internationalized (i18n) and supports the following locales and corresponding languages:

| Locale | Language | Description |
|---|---|---|
| en | English | English locale and language settings (this is the default) |
| it | Italian | Italian locale and language settings |
| es | Spanish | Spanish locale and language settings |
| fr | French | French locale and language settings |
| de | German | German locale and language |
| pl | Polish | Polish locale and language |
| ja | Japanese | Japanese locale and language |
| zh | Mandarin Chinese | Mandarin Chinese locale and language |
| ar | Arabic | Arabic locale and language |
| pt | Portuguese | Portuguese locale and language |
| tr | Turkish | Turkish locale and language |
| ru | Russian | Russian locale and language |
| hi | Hindu | Hindu locale and language |
| bg | Bulgarian | Bulgarian locale and language |
| nl | Dutch | Dutch locale and language |
| is | Icelandic | Icelandic locale and language |
| id | Indonesian | Indonesian locale and language |
| ms | Malay | Malay locale and language |
| th | Thai | Thai locale and language |
| cs | Czech | Czech locale and language |
| hr | Croatian | Croatian locale and language |
| sl | Slovenian | Slovenian locale and language |
| fi | Finnish | Finnish locale and language |
| no | Norwegian | Norwegian locale and language |
| da | Danish | Danish locale and language |
| sv | Swedish | Swedish locale and language |

Note that the English locale is the default, and if you don't specify another locale in the WPF API, this is the locale and language that the web payment form will be translated into.

The customer has the option to change the language once he has been redirected to the WPF via the received `redirect_url`.

It is a good practice to submit the desired locale in the WPF API create call, so that a proper `redirect_url` is returned instead of manually parsing and generating a locale-specific `redirect_url`.

If a locale/language different than the current ones is needed or any translation errors/inconsistencies are spotted, feel free to contact the IT Support team at tech-support@e-comprocessing.com and contribute to the translation effort.

# 3DSecure

3DSecure (3-domain structure), also known as a payer authentication, is a security protocol that helps to prevent fraud in online credit and debit card transactions. This additional security was initiated and created by **Visa** and **MasterCard** and it's branded as **Verified by Visa** and **MasterCard SecureCode** respectively.

**3DSecure (3DS)**: A set of protocols and procedures through which a customer's ownership of the credit card they are using can be assessed. 3DSv1 requires interaction with the customers browser for authentication. 3DSv2 offers a number of enhancements, including frictionless flows.

# V1

#### INTRODUCTION

3DSv1 (referred to as 3D Payer Authentication on the gateway), is a security protocol created by EMVCo and major card schemes that aims to prevent fraudulent use of credit cards by authenticating cardholders in card-not-present (CNP) transactions.

The asynchronous workflow may take place on the merchants site in case the merchant has a MPI or wants to use another MPI. In this case a 3DSecure transaction will be handled synchronous but requires a different set of parameters. See Authorize3d, Sale3d, or InitRecurringSale3d transactions for details.

If the card is considered valid for 3DSecure, the payment becomes asynchronous as the consumer gets redirected to a form at the gateway where he interacts with the issuer to complete the authentication.

The merchant will be notified of the outcome via the `notification_url`.

Both `return_success_url`, `return_failure_url` and `notification_url` need to be supplied in the transaction request, so the the can be properly redirected to a return URL after the payment has taken place. An asynchronous notification is then sent to the `notification_url`.

#### AUTHENTICATION FLOW

> Transactions can fail prior to invoking the 3D mechanism (in case of e.g. invalid card number, expiry date or risk checks). In this case, the 3D transaction becomes synchronous.

> If the card is not participating in 3DS and non-participating cards are allowed on the MID configuration, the async 3D transaction behaves like synchronous and proceeds with the authorization message.

> Please, use only HTTP **GET** method in order to navigate consumer to any of the redirect URLs in the 3DS authentication scope. The other HTTP methods **POST**, **PUT**, etc are not supported for redirect URLs and won't be handled.

> In order to enforce using the 3DSv1 authentication protocol, make sure to **NOT** include the **threeds_v2_params** in the transaction request.

In case the merchant does not have a MPI, the workflow is handled by Genesis and looks like in the diagram below:

Enrolled?
NO
YES
Authorization allowed?
NO
YES
Merchant
<payment_response>
...
<status>error / declined</status>
...
</payment_response>
Merchant
Enqueue
an asynchronous notification
about the payment
to notification_url
(HTTP POST)
Process
Transaction
Gateway
Transaction successful?
NO
YES
Merchant
<payment_response>
...
<status>declined</status>
...
</payment_response>
Merchant
<payment_response>
...
<status>approved</status>
...
</payment_response>
Merchant
Enqueue
an asynchronous notification
about the payment
to notification_url
(HTTP POST)
Payer Authentication
required
Merchant
<payment_response>
...
<status>pending_async</status>
<redirect_url>3ds_v1_payer_authentication_url</...>
<redirect_url_type>3ds_v1_payer_authentication</...>
...
</payment_response>
Gateway
Handle response
Merchant
Redirect consumer to
redirect_url
(HTTP GET)
Gateway
Acquire
Payer authentication
Form
(HTTP POST)
Access Control
Server (ACS)
Authentication Form
provided

Consumer
Render Authentication form
Gateway
Complete the 2FA
PAYER AUTHENTICATION 3DSv1
Consumer
Submit Authentication form (HTTP POST)
Access Control Server (ACS)
Verify Authentication
Access Control Server (ACS)
Callback (HTTP POST)
Gateway
Acquire MPI params
3DS Server
MPI params provided
Gateway
Store MPI params
Authenticated?
NO
YES
Redirect consumer to return_failure_url (HTTP GET)
Consumer
Back to merchant
Merchant
Process Transaction
Enqueue an asynchronous notification about the payment to notification_url (HTTP POST)
Merchant
Gateway

#### TESTING

Test Cards The following test cards can be used to return specific responses in each state

| Card Number | Card Brand | Transaction Result |
|---|---|---|
| 4711100000000000 | Visa | 3DSecure enrolled |
| 4012001037461114 | Visa | 3DSecure enrolled failing authentication |
| 4012001036853337 | Visa | 3DSecure unavailable - Card Not Participating |
| 4012001037484447 | Visa | Error in 3DSecure Network in first step of 3DSecure authentication process |
| 4012001036273338 | Visa | Error in 3DSecure Network in second (asynchronous) step of 3DSecure authentication process |
| 5555555555559997 | MasterCard | 3DSecure enrolled |
| 5185540810000019 | MasterCard | 3DSecure enrolled failing authentication |
| 5454545454545454 | MasterCard | 3DSecure unavailable - Card Not Participating |
| 5200828282828210 | MasterCard | Error in 3DSecure Network in first step of 3DSecure authentication process |
| 5204230080000017 | MasterCard | Error in 3DSecure Network in second (asynchronous) step of 3DSecure authentication process |

When redirected to the dummy authentication page you may enter any password you like.

---

## V2

#### INTRODUCTION

How does 3DS 2.0 work? Better, stronger fraud-detection intelligence, to put it simply.

3DSecure has been around for years and creates an authentication data connection between digital merchants, payment networks and financial institutions to be able to analyze and share more intelligence about transactions. The new 2.0 version of the technology enables a real-time, secure, information-sharing pipeline that merchants can use to send an unprecedented numbe of transaction attributes that the issuer can use to authenticate customers more accurately without asking for a static password or slowing down commerce.

All in all, the second iteration of the 3DSecure protocol is a great improvement for all parties involved.

It allows merchants to provide protection across multiple platforms with easy integration into their systems, including mobile applications, while still being able to exploit the benefits that the protocol provides. It is also estimated that cart abandonment rates will dramatically fall.

Issuers can share and receive more data with merchants, giving them a greater insight into transactional patterns which will allow them to determine the risk with higher accuracy and therefore improve the authentication process. 3DS 2.0 will also offer banks the opportunity to effortlessly comply with the requirements of PSD2.

For customers, the updates are perhaps most beneficial. They can now enjoy protection from fraudulent transactions across most platforms.

Not only will transactions be more secure through the implementation of increased protection methods such as two-factor authentication (2FA), but the consumer experience will also be greatly improved by frictionless flows through risk-based authentication.

3DS 2.0 is an updated protocol created to recognise the need of current and future requirements including adding the support of mobile-based authentication and digital wallets integration. One of the main benefits is the introduction of a frictionless flow that allows issuers to approve a transaction without requiring any manual input from the cardholder.

**3DSecure Challenge**: The 3DS 2.0 step during which the customer interacts with the card issuer to provide additional authentication for the transaction.

**3DSecure Method**: Refers to the 3DS 2.0 step during which the card issuer gathers information about the customer via their browser. A card issuer may use this data to help make a risk-based decision , for example if the issuer can identify that the cardholder has used this web browser to make purchases in the past, they may decide the transaction is low risk and consider the customer authenticated without requiring a 3DSecure Challenge

**3DSecure Method Handling**: The 3DSecure Method allows for additional browser information to be gathered by an ACS prior to receipt of the AReq message to help facilitate the transaction risk assessment. The use of the 3DSecure Method by an ACS is optional. The inclusion of 3DSecure Method URL and account ranges in a Directory Server is optional for an ACS.

**3DSecure 1 Fallback** A possible alternate workflow if certain criteria are met during 3DSecure Authentication Request phase and customer authentication using 3DSecure version 1 is to be used rather than 3D-Secure version 2.

**Access Control Server**: The ACS contains the authentication rules and is controlled by the Issuer. ACS functions include:

- Verifying whether a card number is eligible for 3DSecure authentication
- Verifying whether a Consumer Device type is eligible for 3DSecure authentication
- Authenticating the Cardholder for a specific transaction

**Consumer Device**: The Consumer Device has the capability to run a 3DSecure Requestor App or present a website on a browser that can be used for 3DSecure authentication. The Consumer Device-based components of the 3DSecure Requestor Environment depend on the model:

- App-based - the 3DSecure SDK integrated with the 3DSecure Requestor App
- Browser-based - a browser utilising the 3DSecure Method

**Frictionless**: No consumer interaction with the issuer, the authentication completes

#### REQUEST PARAMS

In order to enforce the 3DSecure v2 authentication protocol, the request params below must be submitted in the transaction request forAuthorize3d Sale3d InitRecurringSale3d.

The **usage** param needs to be supplied when processing asynchronous 3DSecure transaction using the 3DSv2 authentication protocol.

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| **threeds_v2_params** | required* | | 3DSv2 async parameters. They must be submitted in order to use the 3DSv2 authentication protocol in asynchronous workflow |
| **control** | required* | | General params for preferences in authentication flow and providing device interface information. |
| device_type | required* | string | Identifies the device channel of the consumer, **required** in the 3DSv2 authentication protocol. For more information, go to 3DSv2 control params |
| challenge_window_size | required* | string | Identifies the size of the challenge window for the consumer. For more information, go to3DSv2 control params |
| challenge_indicator | optional | string | The value has weight and might impact the decision whether a challenge will be required for the transaction or not. If not provided, it will be interpreted as **no_preference**. For more information, go to3DSv2 control params |
| **purchase** | required* | | Purchase related params providing with additional information regarding the order. |
| category | required* | string | Identifies the type of transaction being authenticated. This field is required in some markets.Accepted values are: **goods**, **service**, **check_acceptance**, **account_funding**, **quasi_cash**, **prepaid_activation**, **loan**. |
| **merchant_risk** | recommended | | Merchant risk assessment params. They are all optional, but recommended' |
| shipping_indicator | optional | string(16) | Indicator code that most accurately describes the shipping method for the cardholder specific transaction. If one or more items are included in the sale, use the Shipping Indicator code for the physical goods. If all digital goods, use the code that describes the most expensive item. Accepted values are: **same_as_billing**, **stored_address**, **verified_address**, **pick_up**, **digital_goods**, **travel**, **event_tickets**, **other**. |
| delivery_timeframe | optional | string(11) | Indicates the merchandise delivery timeframe. Accepted values are: **electronic**, **same_day**, **over_night**, **another_day**. |
| reorder_items_indicator | optional | string(10) | Indicates whether the cardholder is reordering previously purchased merchandise. Accepted values are: **first_time**, **reordered**. |
| pre_order_purchase_indicator | optional | string(21) | Indicates whether cardholder is placing an order for merchandise with a future-availability or release date. Accepted values are: **merchandise_available**, **future_availability**. |
| pre_order_date | optional | dd-mm-yyyy | For a pre-ordered purchase, the expected date that the merchandise will be available. |
| gift_card | optional | 'true' | Prepaid or gift card purchase. |
| gift_card_count | optional | integer | For prepaid or gift card purchase, total count of individual prepaid or gift cards/codes purchased. The value is limited to **99**. |
| **card_holder_account** | recommended | | Cardholder account additional information. They are all optional, but recommended, because they have a significant impact on approval rates |
| creation_date | optional | dd-mm-yyyy | Date that the cardholder opened the account with the 3DS Requester. |
| update_indicator | optional | string(19) | Length of time since the cardholder's account information with the 3DS Requestor was last changed. Includes Billing or Shipping address, new payment account, or new user(s) added. Accepted values are: **current_transaction**, **less_than_30days**, **30_to_60_days**, **more_than_60days**. |
| last_change_date | optional | dd-mm-yyyy | Date that the cardholder's account with the 3DS Requestor was last changed. Including Billing or Shipping address, new payment accout, or new user(s) added. |

| Parameter | Required | Format | Description |
|---|---|---|---|
| password_change_indicator | optional | string(18) | Length of time since the cardholder account with the 3DS Requestor had a password change or account reset. Accepted values are: **no_change**, **during_transaction**, **less_than_30days**, **30_to_60_days**, **more_than_60days**. |
| password_change_date | optional | dd-mm-yyyy | Date that cardholder's account with the 3DS Requestor had a password change or account reset. |
| shipping_address_usage_indicator | optional | string(19) | Indicates when the shipping address used for this transaction was first used with the 3DS Requestor. Accepted values are: **current_transaction**, **less_than_30days**, **30_to_60_days**, **more_than_60days**. |
| shipping_address_date_first_used | optional | dd-mm-yyyy | Date when the shipping address used for this transaction was first used with the 3DS Requestor. |
| transactions_activity_last_24_hours | optional | integer | Number of transactions (successful and abandoned) for this cardholder account with the 3DS Requestor across all payment accounts in the previous 24 hours. |
| transactions_activity_previous_year | optional | integer | Number of transactions (successful and abandoned) for this cardholder account with the 3DS Requestor across all payment accounts in the previous year. |
| provision_attempts_last_24_hours | optional | integer | Number of Add Card attempts in the last 24 hours. |
| purchases_count_last_6_months | optional | integer | Number of purchases with this cardholder account during the previous six months. |
| suspicious_activity_indicator | optional | string(22) | Indicates whether the 3DS Requestor has experienced suspicious activity (including previous fraud) on the cardholder account. Accepted values are: **no_suspicious_observed**, **suspicious_observed**. |
| registration_indicator | optional | string(19) | Indicates the length of time that the payment account was enrolled in the cardholder's account with the 3DS Requester. Accepted values are: **guest_checkout**, **current_transaction**, **less_than_30days**, **30_to_60_days**, **more_than_60days**. |
| registration_date | optional | dd-mm-yyyy | Date that the payment account was enrolled in the cardholder's account with the 3DS Requestor. |
| **browser** | required* | | For browser-based transactions. They are all *required* in case the **device_type** is set to **browser** |
| accept_header | required* | string(2048) | Exact content of the HTTP accept headers as sent to the 3DS Requester from the Cardholder browser. |
| java_enabled | required* | boolean | Boolean that represents the ability of the cardholder browser to execute Java. The value can be retrieved by accessing a property of the navigator with JavaScript, `navigator.javaEnabled`. |
| language | required* | string(8) | Value representing the browser language as defined in IETF BCP47. The value can be retrieved by accessing a property of the navigator with JavaScript, `navigator.language`. |
| color_depth | required* | integer | Value representing the bit depth of the colour palette for displaying images, in bits per pixel. Obtained from Cardholder browser using the `screen.colorDepth` property. The value can be one of **1**, **4**, **8**, **15**, **16**, **24**, **32**, **48** |
| screen_height | required* | integer | Total height of the Cardholder's screen in pixels. Value is returned from the screen.height property. |
| screen_width | required* | integer | Total width of the Cardholder's screen in pixels. Value is returned from the screen.width property. |
| time_zone_offset | required* | string(5) | Time difference between UTC time and the Cardholder browser local time, in minutes. The value can be retrieved using Javascript `getTimezoneOffset()` method. |
| user_agent | required* | string(2048) | Exact content of the HTTP user-agent header. |
| **sdk** | required* | | For application-based transactions. They are all *required* in case the **device_type** is set to **application** |
| interface | required* | string(6) | SDK Interface types that the device of the consumer supports for displaying specific challenge interfaces within the SDK. Accepted values are: **native**, **html**, **both**. |
| **ui_types** | required* | | Lists all UI types that the device of the consumer supports for displaying specific challenge interfaces within the SDK. |
| ui_type | required* | string(13) | UI type that the device of the consumer supports for displaying specific challenge interface. Accepted values are: **text**, **single_select**, **multi_select**, **out_of_bag**, **other_html**. |
| application_id | required* | string(36) | Universally unique ID created upon all installations and updates of the 3DS Requestor APp on a Customer Device. This will be newly generated and stored by the 3DS SDK for each installation or update. The field is limited to 36 characters and it shall have a canonical format as defined in IETF RFC 4122. |
| encrypted_data | required* | string(64000) | JWE Object as defined Section 6.2.2.1 containing data encrypted by the SDK for the DS to decrypt. The data will be present when sending to DS, but not present from DS to ACS. |
| ephemeral_public_key_pair | required* | string(256) | Public key component of the ephemeral key pair generated by the 3DS SDK and used to establish session keys between the 3DS SDK and ACS. In AReq, this data element is contained within the ACS Signed Content JWS Object. The field is limited to maximum 256 characters. |
| max_timeout | required* | integer | Indicates the maximum amount of time (in minutes) for all exchanges. The field shall have value greater or equals than 05. |
| reference_number | required* | string(32) | Identifies the vendor and version of the 3DS SDK that is integrated in a 3DS Requestor App, assigned by EMVCo when the 3DS SDK is approved. The field is limited to 32 characters. |
| **recurring** | required* | | Additional recurring details. |
| expiration_date | required* | dd-mm-yyyy | A future date indicating the end date for any further subsequent transactions. For more information, go to 3DSv2 recurring params |
| frequency | required* | integer | Indicates the minimum number of days between subsequent transactions. For more information, go to3DSv2 recurring params |

`required* = conditionally required`

#### CONTROL PARAMS

`challenge_indicator` - *optional*

| Value | Description |
|---|---|
| **no_preference** *(default)* | Don't have any preferences related to the Challenge flow |
| **no_challenge_requested** | I prefer that a Challenge flow does not take place |
| **preference** | A request for the Challenge flow to take place |
| **mandate** | A Challenge flow must take place to fulfill a mandate |

`device_type` - *required*

| Value | Description |
|---|---|
| **browser** | Browser-based consumer interface. All the `browser` parameters are required. |
| **application** | Application-based consumer interface, mobile etc. All the `sdk` parameters are required. |

`challenge_window_size` - *required**

> The **challenge_window_size** is required when the **device_type** is set to **browser**

**Notes:** *Dimensions of the challenge window that has been displayed to the Cardholder. The ACS shall reply with content that is formatted to appropriately render in this window to provide the best possible consumer experience. Preconfigured sizes are width X height in pixels of the window displayed in the Cardholder browser window.*

| Value | Description |
|---|---|
| **250x400** | Challenge window: width of `250px` and height of `400px` |
| **390x400** | Challenge window: width of `390px` and height of `400px` |
| **500x600** | Challenge window: width of `500px` and height of `600px` |
| **600x400** | Challenge window: width of `600px` and height of `400px` |
| **full_screen** | Challenge in a full screen |

#### RECURRING PARAMS

> All recurring params below are required for asynchronousInitRecurringSale3d that is about to use the 3DSv2 authentication protocol.

`expiration_date` - *required**

| Value | Description |
|---|---|
| *16-01-2021* | No further subsequent transactions are expected after that date. |

`frequency` - *required**

> The **frequency** can be a positive number, not lower than **1** and not greater than **9999**

| Value | Description |
|---|---|
| **14** | Indicates a minimum number of 14 days between subsequent recurring transactions. |
| **30** | Indicates a minimum number of 30 days *(monthly subscription)* between subsequent recurring transactions. |

#### AUTHENTICATION FLOWS

There are two general workflows for 3DS 2.0, **Frictionless** Flow and **Challenge** Flow, as well as a 3DS1 Fallback Flow for circumstances where 3DS2 is not available or the authentication response gained via 3DS2 is not sufficient for obtaining a liability shift.

> Transactions can fail prior to invoking the 3D mechanism (in case of e.g. invalid card number, expiry date or risk checks). In this case, the 3D transaction becomes synchronous.

If the card is not participating in 3DS and non-participating cards are allowed on the MID configuration, the async 3D transaction behaves like synchronous and proceeds with the authorization message.

Please, use only HTTP **GET** method in order to navigate consumer to any of the redirect URLs in the 3DS authentication scope. The other HTTP methods **POST**, **PUT**, etc are not supported for redirect URLs and won't be handled.

#### FRICTIONLESS

A Frictionless Flow occurs when the Issuer authenticates the cardholder without cardholder involvement by evaluating the transaction's risk level.

The customer has more frictionless experience through the merchant's platform by not being challenged. That means the drop-off rate due to the 3DSecure protocol will be drastically reduced and the customer will happily come back to the merchant's platform.

The Frictionless Flow does not require further Cardholder interaction to achieve a successful authentication and complete the 3DSecure authentication process. The payment will be completely synchronous, the consumer won't be redirected to the Access Control Server to complete the authentication.

In order to enforce using the 3DSv2 authentication protocol, make sure to **include** the **threeds_v2_params** in the transaction request.

1.) The merchant submit a 3D transaction to the API including the3DSecure v2 params

2.) No 3DSecure Method or 3DSecure Challenge is required by the card issuer, the transaction will be completed without further action.

3.) A synchronous payment response is returned to the merchant indicating the status of the payment.

The **usage** param needs to be supplied when processing asynchronous 3DSecure transaction using the 3DSv2 authentication protocol.

**FRICTIONLESS WITH 3DSECURE METHOD**

A Frictionless Flow occurs when the Issuer authenticates the cardholder without cardholder involvement by evaluating the transaction's risk level.

The customer has more frictionless experience through the merchant's platform by not being challenged. That means the drop-off rate due to the 3DSecure protocol will be drastically reduced and the customer will happily come back to the merchant's platform.

The Frictionless Flow does not require further Cardholder interaction to achieve a successful authentication and complete the 3DSecure authentication process. The payment will be

completely synchronous, the consumer won't be redirected to the Access Control Server to complete the authentication.

> In order to enforce using the 3DSv2 authentication protocol, make sure to **include** the **threeds_v2_params** in the transaction request.

This flow is treated frictionless (without consumer interaction with the issuer), but with the only difference that the ACS requires a 3DSecure Method to be submitted before continuing.

Thanks to risk-based authentication performed in the ACS, frictionless flow allows issuers to approve a transaction without the need to interact with the cardholder.

When the customer makes an online purchase they would add an item to their shopping cart, fill out the normal purchase information and then proceed to confirm the purchase.

Details of the purchase including device data, item purchased and value are submitted to the ACS server to determine the authenticity of the cardholder.

The ACS will then screen it with the risk-based elements. If the risk is deemed to be low, the ACS can authenticate the customer passively and not bother them with the extra confirmation.

This is a frictionless process for the customer as it happens behind the scenes. They are directed straight to the purchase confirmation screen, without even knowing that their transaction was screened.

3DS Method can be submitted **synchronously** or **asynchronously** depending on the MID configuration.

**Synchronous 3DS Method submission**

The `redirect_url` from the API response will hold the **3DS Method URL**, where the customer must be redirected to using HTTP **GET** method. This will submit the 3DS Method automatically and once it finishes the payment will be processed without the need of an interaction between customer and issuer. The customer will be redirected to the `return_success_url` or `return_failure_url` depending on the transaction status.

f

> The synchronous handling of the 3DS-Method is simpler than the asynchronous, but it's more risky, because the consumer might not be redirected properly after the 3DS-Method submission. This means if, accidentally, the 3DS-Method fails to be submitted (browser incompatibility, JS errors etc), the Gateway won't be called back and the processing is unable to finish. The exact 3DS-Method content (HTML + JS) will be provided by the ACS to retrieve additional risk details for the consumer's browser, but there is no 100% garantie that the 3DS-Method will be succeessfully submitted on all browser agents.

> However, we encourage using the **asynchronous** 3DS-Method handling. For more more information about how to switch between **synchronous** and **asynchronous**, please contact Tech Support

> The **usage** param needs to be supplied when processing asynchronous 3DSecure transaction using the 3DSv2 authentication protocol.

Enrollment status provided
Enrolled?
NO
YES
Authorization allowed?
NO
YES
Merchant
<payment_response>
...
<status>error | declined</status>
...
</payment_response>
Merchant
Enqueue an asynchronous notification about the payment to notification_url (HTTP POST)
Process Transaction
Gateway
Transaction successful?
NO
YES
Merchant
<payment_response>
...
<status>declined</status>
...
</payment_response>
Merchant
<payment_response>
...
<status>approved</status>
...
</payment_response>
Merchant
Enqueue an asynchronous notification about the payment to notification_url (HTTP POST)
3DS-Method required
Gateway
Merchant
<payment_response>
...
<status>pending_async</status>
<redirect_url>3ds_method_redirect_url</...>
<redirect_url_type>3ds_v2_threeds_method</...>
...
</payment_response>
Handle response
Merchant
Redirect consumer to redirect_url (HTTP GET)
Gateway
Acquire 3DS-Method Content (HTTP POST)

Access Control Server (ACS)
3DS-Method content (HTML + JS) provided
Consumer
Render 3DS-Method content (HTML + JS)
Gateway
Automated 3DS-Method form submission (HTTP POST)
Access Control Server (ACS)
Callback (HTTP POST)
3DS Server
Callback (HTTP POST)
Gateway
AUTOMATED 3DS-METHOD SUBMISSION
Acquire MPI params
3DS Server
MPI params provided
Gateway
Store MPI params
Authenticated?
NO
Redirect consumer to return_failure_url (HTTP GET)
Consumer
Back to merchant
Merchant
YES
Process Transaction
Enqueue an asynchronous notification about the payment to notification_url (HTTP POST)
Merchant

**Asynchronous 3DS Method submission**

A link between the customer's browser and the card issuer must be opened with a hidden iframe. It is used for the card issuer to load JavaScript which gathers device information to be returned to the card issuer. The next step after initiating the iframe, is to redirect the customer to `redirect_url` using **HTTP GET**, which will process the payment and redirect the consumer back to `return_success_url` or `return_failure_url` depending on the status of the payment transaction.

> In order to submit the 3DS-Method appropriately, please, make sure the following is done properly:
>
> - **1st action** - Load the **threeds_method_url** in a hidden iframe as described in the diagram below
> - **2nd action** - Wait up to **10s** for the 3DS-Method submission to reach the final state as described in the diagram and redirect consumer to **redirect_url** to complete the payment.
>
> After the consumer is redirected, there will be a specific timeout (a number of seconds) on the 3DS Server level to wait for the 3DS-Method to complete (if not done yet). This means, the next step, which the consumer needs to take in order to complete the authentication, might be delayed a bit on the 3DS Server level, but this is an expected behaviour. The handling of the 3DS-Method will be done properly on the 3DS Server level, but the next step after the 3DS-Method partly depends on the successful execution of the 3DS-Method. So make sure you are performing the redirect after the 3DS-Method has reach its final state as described below, but do NOT wait for the 3DS-Method completion more than **10s**.

The asynchronous submission of the 3DS-Method might look more complex to handle than the synchronous, but may decrease the risk or potential transaction processing interruption, because the consumer redirection does not depend on a successful 3DS-Method submission. The consumer will be redirected to the redirect_url regardless of the 3DS-Method result. It's a responsibility of the Issuer ACS then to take the appropriate decision how to continue with the authentication.

> The **usage** param needs to be supplied when processing asynchronous 3DSecure transaction using the 3DSv2 authentication protocol.

Merchant
<payment_response>
...
<status>error</status>
...
</payment_response>
NO
Card valid?
YES
Acquire
Enrollment status
3DS Server
Enrollment status
provided
NO
Enrolled?
YES
Merchant
<payment_response>
...
<status>error / declined</status>
...
</payment_response>
NO
Authorization
allowed?
YES
Merchant
Enqueue
an asynchronous notification
about the payment
to notification_url
(HTTP POST)
Process
Transaction
Gateway
Merchant
<payment_response>
...
<status>declined</status>
...
</payment_response>
NO
Transaction
successful?
YES
Merchant
<payment_response>
...
<status>approved</status>
...
</payment_response>
Merchant
Enqueue
an asynchronous notification
about the payment
to notification_url
(HTTP POST)
3DS-Method
required
Merchant
<payment_response>
...
<status>pending_async</status>
<redirect_url>3ds_method_continue_url</...>
<redirect_url_type>3ds_v2_threeds_method</...>
<threeds_method_url> 3ds_method_redirect_url</...>
...
</payment_response>
Gateway

1
1st action:
3DS-Method
submission
2
2nd action
Continue
after
3DS-Method
completes
(wait up to
10s for the
3DS-Method
to reach its
final state)
Load
the threeds_method_url
(invisible to the consumer)
The threeds_method_url needs to be
submitted in an iframe using HTTP GET
<iframe id="..." width="xxx" height="xxx" style="display: none"
src="threeds_method_url">
...
</iframe>
Merchant
(iframe)
Initiate
3DS-Method
Gateway
Acquire
3DS-Method Content
(HTTP POST)
Access Control
Server (ACS)
3DS-Method content
(HTML + JS)
provided
Gateway
Render
3DS-Method content
(HTML + JS)
Merchant
(iframe)
Automatic
3DS-Method form submission
(HTTP POST)
Access Control
Server (ACS)
Callback
(HTTP POST)
3DS Server
Callback
(HTTP POST)
Gateway
Render 200 OK
( no response body )
Merchant
(iframe)
No further action needed
( 3DS-Method completed )
AUTOMATED 3DS-METHOD SUBMISSION
(IFRAME)
Merchant
Redirect consumer to
redirect_url
(HTTP GET)
Gateway

#### CHALLENGE

If the ACS determines that further Cardholder interaction is required to complete the authentication, the Frictionless Flow transitions into the Challenge Flow. For example, a challenge may be necessary because the transaction is deemed high-risk, is above certain thresholds, or requires a higher level of authentication due to country mandates (or regulations).

The Challenge Flow occurs when the issuer assesses the risk of the transaction during the frictionless flow and determines that the transaction requires additional cardholder authentication. The frictionless flow transitions into the challenge flow

> In order to enforce using the 3DSv2 authentication protocol, make sure to**include** the **threeds_v2_params** in the transaction request.

The **usage** param needs to be supplied when processing asynchronous 3DSecure transaction using the 3DSv2 authentication protocol.

Merchant
...
<status>approved</status>
...
</payment_response>
Merchant
Enqueue
an asynchronous notification
about the payment
to notification_url
(HTTP POST)
3DS-Challenge
required
Merchant
<payment_response>
...
<status>pending_async</status>
<redirect_url>3ds_challenge_url</redirect_url>
<redirect_url_type>3ds_v2_challenge</...>
...
</payment_response>
Gateway
Handle response
Merchant
Redirect consumer to
redirect_url
(HTTP GET)
Gateway
Acquire
Challenge Form
(HTTP POST)
Access Control
Server (ACS)
Challenge Form
provided
Consumer
Render Challenge
Gateway
Solve Challenge
(strong authentication)
Consumer
Consumer
interacts with the
issuer to
complete the
strong 3DS
authentication
Access Control
Server (ACS)
Solve Challenge
(strong authentication)
3DS-Challenge
Verify Challenge
Consumer
Render another
Challenge
YES
Challenge
again?
NO

Prepare Callback
(Challenge finished)
Access Control Server (ACS)
Callback
(HTTP POST)
Gateway
Acquire MPI params
3DS Server
MPI params provided
Gateway
Store MPI params
Authenticated?
NO
Redirect consumer to return_failure_url (HTTP GET)
Consumer
Back to merchant
Merchant
YES
Process Transaction
Enqueue an asynchronous notification about the payment to notification_url (HTTP POST)
Merchant
Gateway
Transaction successful?
NO
Redirect consumer to return_failure_url (HTTP GET)
Consumer
Back to merchant
Merchant
YES
Redirect consumer to return_success_url (HTTP GET)
Consumer
Back to merchant
Merchant

#### CHALLENGE WITH 3DSECURE METHOD

If the ACS determines that further Cardholder interaction is required to complete the authentication, the Frictionless Flow transitions into the Challenge Flow. For example, a challenge may be necessary because the transaction is deemed high-risk, is above certain thresholds, or requires a higher level of authentication due to country mandates (or regulations).

The Challenge Flow occurs when the issuer assesses the risk of the transaction during the frictionless flow and determines that the transaction requires additional cardholder authentication. The frictionless flow transitions into the challenge flow

> In order to enforce using the 3DSv2 authentication protocol, make sure to**include** the **threeds_v2_params** in the transaction request.

3DS Method can be submitted**synchronously** or **asynchronously** depending on the MID configuration.

**Synchronous 3DS Method submission**

The `redirect_url` from the API response will hold the**3DS Method URL**, where the customer must be redirected to using HTTP**GET** method. This will submit the 3DS Method automatically and once it finishes the payment will be processed without the need of an interaction between customer and issuer. The customer will be redirected to the `return_success_url` or `return_failure_url` depending on the transaction status.

f

> The synchronous handling of the 3DS-Method is simpler than the asynchronous, but it's more risky, because the consumer might not be redirected properly after the 3DS-Method submission. This means if, accidentally, the 3DS-Method fails to be submitted (browser incompatibility, JS errors etc), the Gateway won't be called back and the processing is unable to finish. The exact 3DS-Method content (HTML + JS) will be provided by the ACS to retrieve additional risk details for the consumer's browser, but there is no 100% garantie that the 3DS-Method will be succeessfully submitted on all browser agents.

> However, we encourage using the **asynchronous** 3DS-Method handling. For more more information about how to switch between**synchronous** and **asynchronous**, please contact
>
> Tech Support

> The **usage** param needs to be supplied when processing asynchronous 3DSecure transaction using the 3DSv2 authentication protocol.

3DS Server
Enrollment status provided
Enrolled?
NO
YES
Authorization allowed?
NO
YES
Merchant
<payment_response>
...
<status>error / declined</status>
...
</payment_response>
Merchant
Enqueue
an asynchronous notification
about the payment
to notification_url
(HTTP POST)
Process Transaction
Gateway
Transaction successful?
NO
YES
Merchant
<payment_response>
...
<status>declined</status>
...
</payment_response>
Merchant
<payment_response>
...
<status>approved</status>
...
</payment_response>
Merchant
Enqueue
an asynchronous notification
about the payment
to notification_url
(HTTP POST)
3DS-Method required
Gateway
Merchant
<payment_response>
...
<status>pending_async</status>
<redirect_url>3ds_method_redirect_url</...>
<redirect_url_type>3ds_v2_threeds_method</...>
...
</payment_response>
Handle response
Merchant
Redirect consumer to
redirect_url
(HTTP GET)
Gateway
Acquire

3DS-Method Content
(HTTP POST)
Access Control Server (ACS)
3DS-Method content
(HTML + JS)
provided
Consumer
Render
3DS-Method content
(HTML + JS)
Gateway
Automated
3DS-Method form submission
(HTTP POST)
Access Control Server (ACS)
Callback
(HTTP POST)
3DS Server
Callback
(HTTP POST)
Gateway
AUTOMATED 3DS-METHOD SUBMISSION
Acquire MPI params
3DS Server
3DS-Challenge
required
Gateway
Acquire
Challenge Form
(HTTP POST)
Access Control Server (ACS)
Challenge Form

provided
Consumer
Render Challenge
Gateway
Solve Challenge
(strong authentication)
Consumer
Consumer
interacts with the
issuer to
complete the
strong 3DS
authentication
Access Control
Server (ACS)
Solve Challenge
(strong authentication)
3DS-Challenge
Verify Challenge
Consumer
Render another
Challenge
YES
Challenge
again?
NO
Prepare Callback
(Challenge finished)
Access Control
Server (ACS)
Callback
(HTTP POST)
Gateway
Acquire MPI params
3DS Server
MPI params
provided
Gateway
Store MPI params

**Asynchronous 3DS Method submission**

A link between the customer's browser and the card issuer must be opened with a hidden iframe. It is used for the card issuer to load JavaScript which gathers device information to be returned to the card issuer. The next step after initiating the iframe, is to redirect the customer to `redirect_url` using **HTTP GET**, which will process the payment and redirect the consumer back to `return_success_url` or `return_failure_url` depending on the status of the payment transaction.

> In order to submit the 3DS-Method appropriately, please, make sure the following is done properly:
>
> - **1st action** - Load the **threeds_method_url** in a hidden iframe as described in the diagram below
> - **2nd action** - Wait up to **10s** for the 3DS-Method submission to reach the final state as described in the diagram and redirect consumer to **redirect_url** to complete the payment.
>
> After the consumer is redirected, there will be a specific timeout (a number of seconds) on the 3DS Server level to wait for the 3DS-Method to complete (if not done yet). This means, the next step, which the consumer needs to take in order to complete the authentication, might be delayed a bit on the 3DS Server level, but this is an expected behaviour. The handling of the 3DS-Method will be done properly on the 3DS Server level, but the next step after the 3DS-Method partly depends on the successful execution of the 3DS-Method. So make sure you are performing the redirect after the 3DS-Method has reach its final state as described below, but do NOT wait for the 3DS-Method completion more than **10s**.

The asynchronous submission of the 3DS-Method might look more complex to handle than the synchronous, but may decrease the risk or potential transaction processing interruption, because the consumer redirection does not depend on a successful 3DS-Method submission. The consumer will be redirected to the redirect_url regardless of the 3DS-Method result. It's a responsibility of the Issuer ACS then to take the appropriate decision how to continue with the authentication.

> The **usage** param needs to be supplied when processing asynchronous 3DSecure transaction using the 3DSv2 authentication protocol.

Merchant
<payment_transaction>
<transaction_type>authorize3d | sale3d |...</...>
...
<usage>...</usage>
...
<return_success_url>...</return_success_url>
<return_failure_url>...</return_failure_url>
<notification_url>...</notification_url>
<threeds_v2_params>
...
</threeds_v2_params>
</payment_transaction>
Gateway
Merchant
<payment_response>
...
<status>error</status>
...
</payment_response>
NO
Card valid?
YES
Acquire
Enrollment status
3DS Server
Enrollment status
provided
NO
Enrolled?
YES
Merchant
<payment_response>
...
<status>error | declined</status>
...
</payment_response>
NO
Authorization
allowed?
YES
Merchant
Enqueue
an asynchronous notification
about the payment
to notification_url
(HTTP POST)
Process
Transaction
Gateway
Merchant
<payment_response>
...
<status>declined</status>
...
</payment_response>
NO
Transaction
successful?
YES
Merchant
<payment_response>
...
<status>approved</status>
...
</payment_response>
Enqueue
an asynchronous notification
about the payment
Merchant

to notification_url
(HTTP POST)
3DS-Method
required
Merchant
<payment_response>
...
<status>pending_async</status>
<redirect_url>3ds_method_continue_url</...>
<redirect_url_type>3ds_v2_threeds_method</...>
<threeds_method_url> 3ds_method_redirect_url</...>
...
</payment_response>
Gateway
1
1st action:
3DS-Method
submission
2
2nd action
Continue
after
3DS-Method
completes
(wait up to
10s for the
3DS-Method
to reach its
final state)
Load
the threeds_method_url
(invisible to the consumer)
The threeds_method_url needs to be
submitted in an iframe using HTTP GET
<iframe id="..." width="xxx" height="xxx" style="display: none"
src="threeds_method_url">
...
</iframe>
Merchant
(iframe)
Initiate
3DS-Method
Gateway
Acquire
3DS-Method Content
(HTTP POST)
Access Control
Server (ACS)
3DS-Method content
(HTML + JS)
provided
Merchant
(iframe)
Render
3DS-Method content
(HTML + JS)
Gateway
Automatic
3DS-Method form submission
(HTTP POST)
Access Control
Server (ACS)
Callback
(HTTP POST)
3DS Server
Callback
(HTTP POST)
Merchant
(iframe)
Render 200 OK
( no response body )
Gateway

No further action needed
( 3DS-Method completed )
AUTOMATED 3DS-METHOD SUBMISSION
(IFRAME)
Merchant
Redirect consumer to
redirect_url
(HTTP GET)
Gateway
Acquire MPI params
3DS Server
3DS-Challenge
required
Gateway
Acquire
Challenge Form
(HTTP POST)
Access Control
Server (ACS)
Challenge Form
provided
Consumer
Render Challenge
Gateway
Solve Challenge
(strong authentication)
Consumer
Consumer
interacts with the
issuer to
complete the
strong 3DS
authentication
Access Control
Server (ACS)
Solve Challenge
(strong authentication)
3DS-Challenge
Verify Challenge
Consumer
Render another
Challenge
YES
Challenge
again?

NO
Prepare Callback
(Challenge finished)
Access Control Server (ACS)
Callback
(HTTP POST)
Gateway
Acquire MPI params
3DS Server
MPI params provided
Gateway
Store MPI params
Authenticated?
NO
Redirect consumer to return_failure_url (HTTP GET)
Consumer
Back to merchant
Merchant
YES
Process Transaction
Enqueue an asynchronous notification about the payment to notification_url (HTTP POST)
Merchant
Gateway
Transaction successful?
NO
Redirect consumer to return_failure_url (HTTP GET)
Consumer
Back to merchant
Merchant
YES
Redirect consumer to return_success_url (HTTP GET)
Consumer
Back to merchant
Merchant

**FALLBACK**

A 3DSecure Fallback Flow is possible for circumstances where 3DSv2 is not available or the authentication response gained via 3DSv2 is not sufficient for obtaining a liability shift. Another possibility for the authentication to be downgraded to 3DSv1 could be a card not-enrolled in the 3DSv2 service, but a a record exists in the 3DSv1 system.

> In order to enforce using the 3DSv2 authentication protocol, make sure to**include** the **threeds_v2_params** in the transaction request.

> The **usage** param needs to be supplied when processing asynchronous 3DSecure transaction using the 3DSv2 authentication protocol.

Merchant
Enqueue
an asynchronous notification
about the payment
to notification_url
(HTTP POST)
Process
Transaction
Gateway
Merchant
<payment_response>
...
<status>declined</status>
...
</payment_response>
NO
Transaction successful?
YES
Merchant
<payment_response>
...
<status>approved</status>
...
</payment_response>
Merchant
Enqueue
an asynchronous notification
about the payment
to notification_url
(HTTP POST)
Payer Authentication
(fallback to 3DSv1)
required
Merchant
<payment_response>
...
<status>pending_async</status>
<redirect_url>3ds_payer_authentication_url</...>
<redirect_url_type>3ds_v1_payer_authentication</...>
...
</payment_response>
Gateway
Handle response
Merchant
Redirect consumer to
redirect_url
(HTTP GET)
Gateway
Acquire
Payer authentication
Form
(HTTP POST)
Access Control
Server (ACS)
Authentication Form
provided
Consumer
Render
Authentication form
Gateway
Complete the 2FA
PAYER AUTHENTICATION
(FALLBACK TO 3DSv1)

Consumer
Submit Authentication form (HTTP POST)
Access Control Server (ACS)
Verify Authentication
Access Control Server (ACS)
Callback (HTTP POST)
Gateway
Acquire MPI params
3DS Server
MPI params provided
Gateway
Store MPI params
Authenticated?
NO
Redirect consumer to return_failure_url (HTTP GET)
Consumer
Back to merchant
Merchant
YES
Process Transaction
Enqueue an asynchronous notification about the payment to notification_url (HTTP POST)
Merchant
Gateway
Transaction successful?
NO
Redirect consumer to return_failure_url (HTTP GET)
Consumer
Back to merchant
Merchant
YES
Redirect consumer to return_success_url (HTTP GET)
Consumer
Back to merchant
Merchant

#### FALLBACK WITH 3DSECURE METHOD

A 3DSecure Fallback Flow is possible for circumstances where 3DSv2 is not available or the authentication response gained via 3DSv2 is not sufficient for obtaining a liability shift. Another possibility for the authentication to be downgraded to 3DSv1 could be a card not-enrolled in the 3DSv2 service, but a a record exists in the 3DSv1 system.

> In order to enforce using the 3DSv2 authentication protocol, make sure to**include** the **threeds_v2_params** in the transaction request.

3DS Method can be submitted**synchronously** or **asynchronously** depending on the MID configuration.

**Synchronous 3DS Method submission**

The `redirect_url` from the API response will hold the**3DS Method URL**, where the customer must be redirected to using HTTP**GET** method. This will submit the 3DS Method automatically and once it finishes the payment will be processed without the need of an interaction between customer and issuer. The customer will be redirected to the `return_success_url` or `return_failure_url` depending on the transaction status.

f

> The synchronous handling of the 3DS-Method is simpler than the asynchronous, but it's more risky, because the consumer might not be redirected properly after the 3DS-Method submission. This means if, accidentally, the 3DS-Method fails to be submitted (browser incompatibility, JS errors etc), the Gateway won't be called back and the processing is unable to finish. The exact 3DS-Method content (HTML + JS) will be provided by the ACS to retrieve additional risk details for the consumer's browser, but there is no 100% garantie that the 3DS-Method will be succeessfully submitted on all browser agents.

> However, we encourage using the **asynchronous** 3DS-Method handling. For more more information about how to switch between**synchronous** and **asynchronous**, please contact
>
> Tech Support

> The **usage** param needs to be supplied when processing asynchronous 3DSecure transaction using the 3DSv2 authentication protocol.

3DS Server
Enrollment status provided
Enrolled?
NO
YES
Authorization allowed?
NO
YES
Merchant
<payment_response>
...
<status>error / declined</status>
...
</payment_response>
Merchant
Enqueue
an asynchronous notification
about the payment
to notification_url
(HTTP POST)
Process Transaction
Gateway
Transaction successful?
NO
YES
Merchant
<payment_response>
...
<status>declined</status>
...
</payment_response>
Merchant
<payment_response>
...
<status>approved</status>
...
</payment_response>
Merchant
Enqueue
an asynchronous notification
about the payment
to notification_url
(HTTP POST)
3DS-Method required
Merchant
<payment_response>
...
<status>pending_async</status>
<redirect_url>3ds_method_redirect_url</...>
<redirect_url_type>3ds_v2_threeds_method</...>
...
</payment_response>
Gateway
Handle response
Merchant
Redirect consumer to
redirect_url
(HTTP GET)
Gateway
Acquire

3DS-Method Content
(HTTP POST)
Access Control Server (ACS)
3DS-Method content
(HTML + JS)
provided
Consumer
Render
3DS-Method content
(HTML + JS)
Gateway
Automated
3DS-Method form submission
(HTTP POST)
Access Control Server (ACS)
Callback
(HTTP POST)
3DS Server
Callback
(HTTP POST)
Gateway
AUTOMATED 3DS-METHOD SUBMISSION
Acquire MPI params
3DS Server
Payer Authentication
(fallback to 3DSv1)
required
Gateway
Acquire
Payer authentication
Form
(HTTP POST)
Access Control Server (ACS)

Authentication Form provided
Consumer
Render Authentication form
Gateway
Complete the 2FA
PAYER AUTHENTICATION (FALLBACK TO 3DSv1)
Consumer
Submit Authentication form (HTTP POST)
Access Control Server (ACS)
Verify Authentication
Access Control Server (ACS)
Callback (HTTP POST)
Gateway
Acquire MPI params
3DS Server
MPI params provided
Gateway
Store MPI params
Merchant
Back to merchant
Consumer
Redirect consumer to return_failure_url (HTTP GET)
NO
Authenticated?
YES
Merchant
Enqueue an asynchronous notification about the payment to notification_url (HTTP POST)
Process Transaction

**Asynchronous 3DS Method submission**

A link between the customer's browser and the card issuer must be opened with a hidden iframe. It is used for the card issuer to load JavaScript which gathers device information to be returned to the card issuer. The next step after initiating the iframe, is to redirect the customer to `redirect_url` using **HTTP GET**, which will process the payment and redirect the consumer back to `return_success_url` or `return_failure_url` depending on the status of the payment transaction.

> In order to submit the 3DS-Method appropriately, please, make sure the following is done properly:
>
> - **1st action** - Load the **threeds_method_url** in a hidden iframe as described in the diagram below
> - **2nd action** - Wait up to **10s** for the 3DS-Method submission to reach the final state as described in the diagram and redirect consumer to **redirect_url** to complete the payment.
>
> After the consumer is redirected, there will be a specific timeout (a number of seconds) on the 3DS Server level to wait for the 3DS-Method to complete (if not done yet). This means, the next step, which the consumer needs to take in order to complete the authentication, might be delayed a bit on the 3DS Server level, but this is an expected behaviour. The handling of the 3DS-Method will be done properly on the 3DS Server level, but the next step after the 3DS-Method partly depends on the successful execution of the 3DS-Method. So make sure you are performing the redirect after the 3DS-Method has reach its final state as described below, but do NOT wait for the 3DS-Method completion more than **10s**.

The asynchronous submission of the 3DS-Method might look more complex to handle than the synchronous, but may decrease the risk or potential transaction processing interruption, because the consumer redirection does not depend on a successful 3DS-Method submission. The consumer will be redirected to the redirect_url regardless of the 3DS-Method result. It's a responsibility of the Issuer ACS then to take the appropriate decision how to continue with the authentication.

> The **usage** param needs to be supplied when processing asynchronous 3DSecure transaction using the 3DSv2 authentication protocol.

...
</payment_response>
YES
Acquire
Enrollment status
3DS Server
Enrollment status
provided
NO
Enrolled?
YES
Merchant
<payment_response>
...
<status>error / declined</status>
...
</payment_response>
NO
Authorization
allowed?
YES
Merchant
Enqueue
an asynchronous notification
about the payment
to notification_url
(HTTP POST)
Process
Transaction
Gateway
Merchant
<payment_response>
...
<status>declined</status>
...
</payment_response>
NO
Transaction
successful?
YES
Merchant
<payment_response>
...
<status>approved</status>
...
</payment_response>
Merchant
Enqueue
an asynchronous notification
about the payment
to notification_url
(HTTP POST)
3DS-Method
required
Merchant
<payment_response>
...
<status>pending_async</status>
<redirect_url>3ds_method_continue_url</...>
<redirect_url_type>3ds_v2_threeds_method</...>
<threeds_method_url> 3ds_method_redirect_url</...>
...
</payment_response>
Gateway
1

1st action:
3DS-Method
submission
2
2nd action
Continue
after
3DS-Method
completes
(wait up to
10s for the
3DS-Method
to reach its
final state)
Load
the threeds_method_url
(invisible to the consumer)
The threeds_method_url needs to be
submitted in an iframe using HTTP GET
<iframe id="..." width="xxx" height="xxx" style="display: none"
src="threeds_method_url">
...
</iframe>
Merchant
(iframe)
Initiate
3DS-Method
Gateway
Acquire
3DS-Method Content
(HTTP POST)
Access Control
Server (ACS)
3DS-Method content
(HTML + JS)
provided
Merchant
(iframe)
Render
3DS-Method content
(HTML + JS)
Gateway
Automatic
3DS-Method form submission
(HTTP POST)
Access Control
Server (ACS)
Callback
(HTTP POST)
3DS Server
Callback
(HTTP POST)
Merchant
(iframe)
Render 200 OK
( no response body )
Gateway
No further action needed
( 3DS-Method completed )
AUTOMATED 3DS-METHOD SUBMISSION
(IFRAME)
Merchant
Redirect consumer to
redirect_url
(HTTP GET)
Gateway

Acquire MPI params
3DS Server
Payer Authentication
(fallback to 3DSv1)
required
Gateway
Acquire
Payer authentication
Form
(HTTP POST)
Access Control
Server (ACS)
Authentication Form
provided
Consumer
Render
Authentication form
Gateway
Complete the 2FA
PAYER AUTHENTICATION
(FALLBACK TO 3DSv1)
Consumer
Submit
Authentication
form
(HTTP POST)
Access Control
Server (ACS)
Verify
Authentication
Access Control
Server (ACS)
Callback
(HTTP POST)
Gateway
Acquire MPI params
3DS Server

**TESTING**

| Scenario | 3DSecure Method | 3DSecure Challenge | Fallback to 3DSv1 | Result | Card Number |
|---|---|---|---|---|---|
| Frictionless | - | - | - | Authenticated | 4012000000060085 |
| Frictionless | Y | - | - | Authenticated | 4066330000000004 |
| Frictionless | - | - | - | Not authenticated | 4111110000000922 |
| Frictionless | Y | - | - | Not authenticated | 4111112232423922 |
| Challenge | - | Y | - | Choose Challenge result | 4918190000000002 |
| Challenge | Y | Y | - | Choose Challenge result | 4938730000000001 |
| Fallback to 3DSv1 | - | - | Y | Choose PaRes response | 4901170000000003 |
| Fallback to 3DSv1 | Y | - | Y | Choose PaRes response | 4901164281364345 |

# Authentication Services

## Introduction

Authentication Services provide Strong Customer Authentication (SCA) which is a type of authentication relying on two or more independent elements.

SCA as defined by the European Central Bank (ECB) and in the context of the EU's Payment Services Directive (PSD2) is:

"a procedure based on the use of two or more of the following elements categorised as knowledge, ownership and inherence:

- something only the user knows, e.g. static password, code, personal identification number;
- something only the user possesses, e.g.token, smart card, mobile phone;
- something the user is, e.g. biometric characteristic, such as a fingerprint.

In addition, the elements selected must be mutually independent, i.e. the breach of one does not compromise the other(s).

At least one of the elements should be non-reusable and non-replicable (except for inherence), and not capable of being surreptitiously stolen via the Internet.

The strong authentication procedure should be designed in such a way as to protect the confidentiality of the authentication data."

Using multiple solutions from the same category would not constitute SCA.

#### SCA WITH 3DSECURE

Only the newest variant of 3DSecure involving one-time passwords (OTP) constitutes a form of SCA.

3DSecure however might be a weak SCA solution for several reasons:

1. it relies on the cards being actively enrolled by the cardholder after issuing, i.e prior any transaction made with it and
2. it is not available on all card schemes.

#### ALTERNATIVE SERVICES

A number of services exist that provide SCA in a compliant way.

They rely on different solutions like generating payment "secrets", one-time passwords etc. Key advantage of such services is that they allow cards to be enrolled "on-the-fly", i.e. during a transaction rather than having to be pre-enrolled after issuing.

Such services could potentially work with any card and are thus card scheme agnostic.

> Authentication services are not available by default, they need to be enabled on your account. Please contact Tech Support for further assistance.

# Genesis KYC Services

## General Info

Genesis KYC Services gives us the ability to perform particular checks on the integrity of the consumer data. Based on the returned consumer score we can decide whether we want to reject/approve a given transaction or perform another action for this consumer.

> You must contact support in order to obtain KYC service credentials.

---

## Create Consumer Registration

Review all aspects of the customer's information, as it is received in the registration process, against local and external databases to increase accuracy and produce a risk score for that customer.

`POST /kyc_service/create_consumer`

```
curl https://username:password@staging.gate.e-comprocessing.net/kyc_service/create_consumer \
  -X POST \
  -H "Content-Type: application/json" \
  -d '
 {
   "customer_information": {
     "first_name": "John",
     "last_name": "Doe",
     "customer_email": "jdoe@example.com",
     "address1": "Sample Address",
     "address2": "4th floor",
     "city": "MyCity",
     "zip_code": "32132",
     "country": "BG",
     "province": "ProvinceName",
     "phone1": "0883113332",
     "phone2": "0883113334",
     "birth_date": "1987-03-03",
     "document_number": "f2345838972",
     "document_type": 3,
     "gender": "M"
   },
   "customer_unique_id": "21343253",
   "customer_registration_date": "2016-12-12",
   "customer_registration_ip_address": "255.255.255.255",
   "customer_username": "dimo16",
   "customer_registration_device_id": "12343242",
   "third_party_device_id": "3432424",
   "profile_action_type": 1,
   "device_fingerprint_type": 1,
   "current_profile_status": 1,
   "bonus_code": "1922",
   "bonus_amount": 100,
   "merchant_website": "dai.com",
   "how_did_you_hear": "friend",
   "affiliate_id": "1922"
 }'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| session_id | optional | string | If this value is not provided the user email account should be complete and valid |
| customer_information | required | object | Customer information. See below for each of the nested required fields |
| customer_username | optional | string | Username of the customer on your system |
| customer_unique_id | required | string | Unique user identificator on your system |
| customer_registration_date | required | string | Date in which the customer was registered in the system OR the date in which the customer was created in the cashier Database yyyy-mm-dd |
| customer_registration_ip_address | required | string | IP address of customer used when the customer was registered in the system OR the current IP address |
| customer_registration_device_id | optional | string | Propietary DeviceId technology, refer to the DeviceId Instruction Manual (provided on request) |
| third_party_device_id | optional | string | Third Party DeviceId |
| device_fingerprint | optional | string | Open Source DeviceId technologies (Intepreted as a String) |
| device_fingerprint_type | optional | enum | 1 - Custom; 2 - Open Source; 3 - Open Source 2; |
| profile_action_type | optional | enum | 1 - Registration;2 - Profile Update; |
| profile_current_status | optional | enum | 0 - Undefined;1 - Review;2 - Denied;3 - Approved; |
| bonus_code | optional | string | Open text variable. Represents the code entered by the customer |
| bonus_submission_date | optional | string | |
| bonus_amount | optional | number | |
| merchant_website | optional | string | |
| industry_type | optional | string | 1 - Finance; 2 - Gambling; 3 - Crypto; 4 - Travel; 5 - Retail; 6 - Risk Vendor; 7 - Adult; 8 - Remittance/Transfer; 9 - Other; |
| how_did_you_hear | optional | string | |
| affiliate_id | optional | string | |
| rule_context | optional | number | Number assigned to a given rule context. Please contact to get the available contexts |

`required* = conditionally required`

**Customer Information Fields**

The fields of the customer information object.

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| first_name | required | string | Customer first name |
| middle_name | optional | string | |
| last_name | required | string | Customer last name |
| customer_email | required | string | Must contain valid e-mail of customer |
| address1 | required | string | Primary address |
| address2 | optional | string | Secondary address |
| city | required | string | City |
| province | required | string | |
| zip_code | required | string | ZIP code |
| country | required | string | two-letter iso codes |
| phone1 | optional | number | |
| phone2 | optional | number | |
| birth_date | optional | string | Required for Visa only when MCC is a Financial Services one (e.g. MCC 6012) |
| document_type | optional | enum | 0 - SSN; 1 - Passport Registry; 2 - Personal ID / National ID; 3 - Identity Card; 4 - Driver License; 8 - Travel Document; 12 - Residence Permit; 13 - Identity Certificate; 16 - Registro Federal de Contribuyentes; 17 - Credencial de Elector; 18 - CPF |
| document_number | optional | string | |
| gender | optional | enum | F - female; M - male |

`required* = conditionally required`


```
{
  "code": 0,
  "message": "Successful Response",
  "technical_message": "Successful Response",
  "details": [
    {
      "reference_id": "233",
      "risk_score": 98,
      "kyc_provider_recommendation": "Reject",
      "rules_triggered": [
        {
          "name": "Multi-Accounting : IP shared to Chargeback reason",
          "risk_score": "100.00",
          "display_to_merchant": 1
        }
      ],
      "scrubber_results": {
        "geo_check": "",
        "address_verification": "",
        "phone_verify": "",
        "idv_usa": "",
        "idv_global": "",
        "gav": "",
        "idv_br": "",
        "bav_usa": "",
        "bav_advanced": "",
        "cb_aml": "",
        "cb_bvs": "",
        "email_age": "",
        "compliance_watchlist": "",
        "iovation": "",
        "idv_advance": ""
      },
      "result_confidence_level": 91.5
    }
  ]
}
```


**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| code | number | genesis success code - 0 |
| message | string | const 'Successful Response' |
| technical_message | string | const 'Successful Response' |
| **details** | | |
| reference_id | | |
| risk_score | | |

| Parameter | Type | Description |
|---|---|---|
| kyc_provider_recommendation | | |
| result_confidence_level | | |
| ... | | |

---

# Update Consumer Registration

Update the customer registration to be able to pass on the latest status required so we can continue improving the data models and provide the best scores and recommendations possible.

`POST /kyc_service/update_consumer`

```
curl https://username:password@staging.gate.e-comprocessing.net/kyc_service/update_consumer \
  -X POST \
  -H "Content-Type: application/json" \
  -d '
 {
  "reference_id": "9999333344443",
  "profile_current_status": "21343253",
  "status_reason": "All good"
 }'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| reference_id | required | number | Unique id returned by corresponding transaction |
| profile_current_status | required | enum | 0 - Undefined; 1 - Review; 2 - Denied; 3 - Approved; |
| status_reason | optional | string | Required only if status is Reject / Decline / Chargeback / Refund / Return / Void |

`required* = conditionally required`

Make sure that `reference_id` points to a preliminary created transaction (will describe transactions below).

```
 {
  "code": 0,
  "message": "Successful Response",
  "technical_message": "Successful Response",
  "details": [
    {
      "reference_id": "9999333344443"
    }
  ]
 }
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| code | number | genesis success code - 0 |
| message | string | const 'Successful Response' |
| technical_message | string | const 'Successful Response' |
| **details** | | |
| reference_id | | |

---

# Create Transaction

Implement this to scrub a new transaction. We will take the information specific to that transaction and run various verification checks available, returning the recommendation, score, and third-party verification scrubbing results.

`POST /kyc_service/create_transaction`

```
curl https://username:password@staging.gate.e-comprocessing.net/kyc_service/create_transaction \
  -X POST \
  -H "Content-Type: application/json" \
  -d '
 {
   "customer_information": {
     "first_name": "John",
     "last_name": "Doe",
     "customer_email": "jdoe@example.com",
     "address1": "Sample address",
     "city": "Paris",
     "zip_code": "666",
     "country": "FR",
     "province": "MyProvince"
   },
   "deposit_limits": {
     "payment_method": "CC"
   },
   "transaction_unique_id": "1332",
   "payment_details": {
     "bin": "411111",
     "last_digits": "1111"
   },
   "customer_ip_address": "255.255.255.255",
   "transaction_created_at": "2016-12-12 23:23:23",
   "currency": "USD",
   "amount": 99999
 }'
```

```
curl https://username:password@staging.gate.e-comprocessing.net/kyc_service/create_transaction \
  -X POST \
  -H "Content-Type: application/json" \
  -d '
 {
   "customer_information": {
     "first_name": "John",
     "last_name": "Doe",
     "customer_email": "jdoe@example.com",
     "address1": "Sample address",
     "city": "Paris",
     "zip_code": "666",
     "country": "FR",
     "province": "MyProvince"
   },
   "deposit_limits": {
     "payment_method": "EW"
   },
   "transaction_unique_id": "1332",
   "payment_details": {
     "ewallet_id": "411111"
   },
   "customer_ip_address": "255.255.255.255",
   "transaction_created_at": "2016-12-12 23:23:23",
   "currency": "USD",
   "amount": 99999
 }'
```

```
curl https://username:password@staging.gate.e-comprocessing.net/kyc_service/create_transaction \
  -X POST \
  -H "Content-Type: application/json" \
  -d '
 {
   "customer_information": {
     "first_name": "John",
     "last_name": "Doe",
     "customer_email": "jdoe@example.com",
     "address1": "Sample address",
     "city": "Paris",
     "zip_code": "666",
     "country": "FR",
     "province": "MyProvince"
   },
   "deposit_limits": {
     "payment_method": "CC"
   },
   "transaction_unique_id": "1332",
   "payment_details": {
     "bin": "411111",
     "tail": "1111"
   },
   "customer_ip_address": "255.255.255.255",
   "transaction_created_at": "2016-12-12 23:23:23",
   "currency": "USD",
   "amount": 99999
 }'
```

```
curl https://username:password@staging.gate.e-comprocessing.net/kyc_service/create_transaction \
  -X POST \
  -H "Content-Type: application/json" \
  -d '
 {
   "customer_information": {
     "first_name": "John",
     "last_name": "Doe",
     "customer_email": "jdoe@example.com",
     "address1": "Sample address",
     "city": "Paris",
     "zip_code": "666",
     "country": "FR",
     "province": "MyProvince"
   },
   "deposit_limits": {
     "payment_method": "EC"
   },
   "transaction_unique_id": "1332",
   "payment_details": {
     "bin": "411111",
     "tail": "4978784327847832784789"
   },
   "customer_ip_address": "255.255.255.255",
   "transaction_created_at": "2016-12-12 23:23:23",
   "currency": "USD",
   "amount": 99999
 }'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| session_id | optional | string | If this value is not provided the user email account should be complete and valid |
| customer_username | optional | string | Username of the customer on your system |
| customer_unique_id | optional | string | Unique user identificator on your system |
| customer_status | optional | string | Customer loyalty level; for example: VIP; Bronze; Platinum; Gold; etc. This is an open text variable |
| customer_loyalty_level | optional | string | Customer loyalty level; for example: VIP; Bronze; Platinum; Gold; etc. This is an open text variable |
| customer_registration_date | optional | string | Date in which the customer was registered in the system OR the date in which the customer was created in the cashier Database yyyy-mm-dd |
| customer_registration_ip_address | optional | string | IP address of customer used when the customer was registered in the system OR the current IP address |
| customer_registration_device_id | optional | string | Propietary DeviceId technology, refer to the DeviceId Instruction Manual (provided on request) |
| customer_information | required | object | Customer information. See below for each of the nested required fields |
| first_deposit_date | optional | string | Empty if first deposit yyyy-mm-dd |
| first_withdrawal_date | optional | string | Empty if 0 withdrawals yyyy-mm-dd |
| deposits_count | optional | number | |
| withdrawals_count | optional | number | |
| current_balance | optional | number | |
| deposit_limits | required | object | See below |
| transaction_unique_id | required | string | Transaction id |
| billing_information | optional | object | See below |
| shipping_information | optional | object | See below |
| payment_details | required | object | See below |
| amount | optional | number | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| currency | optional | string | ISO 4217 Three digits |
| transaction_created_at | required | string | Represents the time of the transaction on the Merchant server. Format: yyyy-mmdd hh:mm:ss |
| transaction_status | optional | enum | Transaction status; it is recommended to send 0 on the initial call. Afterwards call Update Transaction endpoint to update the status. 0 - numberUndefined; 1 - number- Approved; 2 - number- Pre-Auth; 3 - number- Settled; 4 - number- Void; 5 - number- Rejected internally by Negative Database or other scrubber decided to reject the transaction; 6 - number- Declined the bank / gateway / processor rejected the transaction; 7 - number- Chargeback; 8 - number- Return;9 - number- Pending; 10 - number- Pass Transaction validation;11 - number- Failed Transaction validation; 12 - number- efund; 13 - number- Approved Review; 14 - number- Abandon This status is used when the user just leaves the transaction; |
| customer_ip_address | required | string | Customers IP address |
| customer_device_id | optional | string | Propietary DeviceId technology; refer to the DeviceId Instruction Manual (provided on request) |

| Parameter | Required | Format | Description |
|---|---|---|---|
| third_party_device_id | optional | string | Third Party DeviceId |
| device_fingerprint | optional | string | Open Source DeviceId technologies (Intepreted as a String) |
| device_fingerprint_type | optional | enum | 1 - Custom; 2 - Open Source; 3 - Open Source 2; |
| shopping_cart_items_count | optional | number | Represents the quantity of items in the shopping cart |
| local_time | optional | string | Represents the local time of the customer doing the transaction. Format: yyyy-mmdd hh:mm:ss |
| order_source | optional | enum | internet; mobile; inhouse |
| merchant_website | optional | string | Open text variable; it represents the website name or URL that submitted the transaction |
| industry_type | optional | enum | Definition of the industry type the transaction was performed on; 1-number - Finance; 2-number - Gambling; 3-number - Crypto; -number - Travel; 5-number - Retail; 6-number - Risk Vendor; 7-number - Adult; 8-number - Remittance/Transfer; 9-number - Other; |
| customer_password | optional | string | Open text variable; it represents the customers password in hashed format (using MD5) some companies share that information in order to look for patterns |
| rule_context | optional | number | Number assigned to a given rule context. Please contact to get the available contexts. |
| custom_variable | optional | string | Represents anything the merchant wants to store with this transaction |

`required* = conditionally required`

**Deposit Limits Fields**

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| payment_method | required | enum | CC; EC; EW - CreditCard; Echeck; EWallet |
| minimum | optional | number | Lowest valid amount for deposit in minor currency units; ex: 100 = $1 |
| daily_maximum | optional | number | In minor currency units |
| weekly_maximum | optional | string | In minor currency units |
| monthly_maximum | optional | string | In minor currency units |

`required* = conditionally required`

**Billing Information Fields**

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| first_name | optional | string | Customer first name |
| last_name | optional | string | Customer last name |
| customer_email | optional | string | Must contain valid e-mail of customer |
| address1 | optional | string | Primary address |
| address2 | optional | string | Secondary address |
| city | optional | string | City |
| province | optional | string | |
| zip_code | optional | string | ZIP code |
| country | optional | string | ISO 3166-1 Alpha-2. For example: USD |
| phone1 | optional | number | Numbers only; no dash or any other separator. Please include area code if applicable. Country code is not required |
| birth_date | optional | string | yyyy-mm-dd |
| gender | optional | enum | M; F |

`required* = conditionally required`

**Shipping Information Fields**

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| first_name | optional | string | Customer first name |

| Parameter | Required | Format | Description |
|---|---|---|---|
| last_name | optional | string | Customer last name |
| customer_email | optional | string | Must contain valid e-mail of customer |
| address1 | optional | string | Primary address |
| address2 | optional | string | Secondary address |
| city | optional | string | City |
| province | optional | string | |
| zip_code | optional | string | ZIP code |
| country | optional | string | ISO 3166-1 Alpha-2. For example: USD |
| phone1 | optional | number | Numbers only; no dash or any other separator. Please include area code if applicable. Country code is not required |

`required* = conditionally required`

**Payment Details Fields**

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| bin | optional | string | First 6 digits of the card number; Only required if Payment Method Type is Credit Card; |
| tail | optional | string | Last 4 digits of the card number; Only required if Payment Method Type is Credit Card; |
| cvv_present | optional | string | Indicator if the CVV was received or not; The expected values in this field are Yes or No; |
| hashed_pan | optional | string | Only required if Payment Method Type is Credit Card; It should be hashed using SHA256; the string to be hashed is Card Numberand the MD5 hash of the Expiration Date; For example; if the card number is 1111-2222-3333-4444 with expiration date 01/22; The have should be done on the string without spaces nor dash nor other special chars; The MD5 of the Expiration Date 0122 is f0f4b6598f2cee45644673998b4f44be; That said; the string to be hashed is 1111222233334444f0f4b6598f2cee45644673998b4f44be which generates the following result fecd244b7d647b0db391e35910e0d42aaf88f7633a4f4f4883b109abad1d6d7 |
| routing | optional | string | Routing number; Only required if Payment Method Type is eCheck; |
| account | optional | string | Only numbers up to 30 digits; Only required if Payment Method Type is eCheck; |
| ewallet_id | optional | string | Most of the times its an email; Only required if Payment Method Type is eWallet; |

`required* = conditionally required`

Required attributes, depending on the Payment Method:

- For payment method CC
  - bin
  - tail
- For payment method CC OPTIONAL
  - cvv_present
  - hashed_pan
- For payment method EC
  - routing
  - account
- For payment method EW
  - ewallet_id

```
{
  "code": 0,
  "message": "Successful Response",
  "technical_message": "Successful Response",
  "details": [
    {
      "reference_id": "481119",
      "kyc_provider_recommendation": "Approve",
      "risk_score": 0.39,
      "third_party": "",
      "processors": "",
      "reason": "",
      "result_confidence_level": 91.5,
      "rules_triggered": [
        {
          "name": "Address Verification : Global 25",
          "score": "0",
          "display_to_merchant": 1
        }
      ],
      "bin_information": {
        "bank_name": "BANCO NACIONAL DE COSTA RICA",
        "bank_location": "COSTA RICA",
        "card_type": "STANDARD",
        "card_level": "DEBIT",
        "iso_card_country": "CR",
        "card_brand_db": "MASTERCARD",
        "card_brand_script": "MASTERCARD"
      },
      "scrubber_results": {
        "geo_check": "",
        "address_verification": "",
        "phone_verify": "",
        "idv_usa": "",
        "idv_global": "",
        "gav": "",
        "idv_br": "",
        "bav_usa": "",
        "bav_advanced": "",
        "cb_aml": "",
        "cb_bvs": "",
        "email_age": "",
        "compliance_watchlist": "",
        "iovation": "",
        "idv_advance": ""
      }
    }
  ]
}
```


**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| code | number | genesis success code - 0 |
| message | string | const 'Successful Response' |
| technical_message | string | const 'Successful Response' |
| **details** | | |
| reference_id | | |
| kyc_provider_recommendation | | |
| risk_score | | |
| result_confidence_level | | |
| ... | | |

---

# Update Transaction

Utilize this method to update a particular transaction status so we can continue improving the data models and provide the best scores and recommendations.

`POST /kyc_service/update_transaction`

```
curl https://username:password@staging.gate.e-comprocessing.net/kyc_service/update_transaction \
  -X POST \
  -H "Content-Type: application/json" \
  -d '
 {
   "transaction_status": 1,
   "reference_id": "4454982",
   "reason": "a reason message",
   "transaction_unique_id": "1332",
   "cvv_check_result": "",
   "avs_check_result": "",
   "processor_identifier": ""
 }'
```

```
curl https://username:password@staging.gate.e-comprocessing.net/kyc_service/update_transaction \
  -X POST \
  -H "Content-Type: application/json" \
  -d '
 {
   "transaction_status": 2,
   "transaction_unique_id": "1332"
 }'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| session_id | optional | string | If this value is not provided the user email account should be complete and valid |
| transaction_unique_id | required | string | Transaction id |
| reference_id | optional | number | Required only if status is Reject / Decline / Chargeback / Refund / Return / Void |
| transaction_status | optional | enum (number) | 1-Approved; 2-Pre-Auth; 3-Settled; 4-Void; 5-Rejected internally by Negative Database or other scrubber decided to reject thetransaction; 6-Declined the bank / gateway / processor rejected the transaction; 7-Chargeback; 8-Return; 9-Pending; 10-Pass Transaction validation; 11-Failed ransaction validation; 12-Refund; 13-Approved Review; 14-Abandon This status is used when the user just leaves the transaction; |
| reason | optional | string | Required only if status is Reject / Decline / Chargeback / Refund / Return / Void |
| cvv_check_result | optional | string | Response from processor regarding CVV check |
| avs_check_result | optional | string | Result from processor regarding AVS check |
| processor_identifier | optional | string | Unique identifier of the processor attempted |

`required* = conditionally required`

```
 {
  "code": 0,
  "message": "Successful Response",
  "technical_message": "Successful Response",
  "details": [
    {
      "reference_id": "481119"
    }
  ]
 }
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| code | number | genesis success code - 0 |
| message | string | const 'Successful Response' |
| technical_message | string | const 'Successful Response' |
| **details** | | |
| reference_id | | |

---

# Identity Document Upload

Used to verify documents provided by the customer.

When called this action returns the following:

- An instant response with a reference id and additional keys by which the particular query could be addressed.
- Async responses with answers from the performed checks (OCR, manual, etc.). Usually, this responses arrive after 5-7 minutes. The notification url is agreed on environment basis. Create a new transaction for the particular user before making a call of this kind.

`POST /kyc_service/upload_document`

```
curl https://username:password@staging.gate.e-comprocessing.net/kyc_service/upload_document \
  -X 3 \
  -H "Content-Type: application/json" \
  -d '
 {
  "transaction_unique_id": "1332",
  "doc": {
    "mime_type": "image/jpeg",
    "base64_content": "{base_64_endcoded_content}"
  }
 }'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| customer_username | optional | string | Username of the customer on your system |
| customer_unique_id | optional | string | Unique user identificator on your system |
| transaction_unique_id | required* | string | Unique Transaction Id with info of the customer to be verified. Please note; if Transaction Id and Customer Registration Id are rovided the system will use the Transaction Id. Please provide the Transaction Id or the Customer Registration Id; one of them must be provided |
| reference_id | required* | string | Unique Customer Registration Id with info of the customer to be verified |
| method | required | enum (number) | 1 - Manual; 2 - OCR; 3 - Both; |
| doc | required | object | see below |
| doc2; doc3; doc4 | optional | object | additional document images |

`required* = conditionally required`

One of `transaction_unique_id` and `reference_id` fields is required.

**Document Fields**

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| base64_content | required | string | |
| mime_type | required | string | |

`required* = conditionally required`

```
 {
  "code": 0,
  "message": "Successful Response",
  "technical_message": "Successful Response",
  "details": [
    {
      "kyc_source": "OCR",
      "reference_id": "382"
    },
    {
      "kyc_source": "Semi-manual",
      "reference_id": "129"
    }
  ],
  "doc": "51f7e411cd1202e040b21655738beb89",
  "doc2": "b0139d3fb1785c1e57ac5a6b054692eb",
  "doc3": "e0059153192876ce8ab36d8cd8ab1bca"
 }
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| code | number | genesis success code - 0 |
| message | string | const 'Successful Response' |
| technical_message | string | const 'Successful Response' |
| **details** | | |
| kyc_source | | |
| reference_id | | |

# Identity Document Download

Uploaded documents will be stored by legal provisions and they can be requested for review. Just post a JSON body with the identity document id of the given document and a response with the filename and the base64 encoded content of that file would be returned.

`POST /kyc_service/download_document`

```
curl https://username:password@staging.gate.e-comprocessing.net/kyc_service/download_document \
  -X POST \
  -H "Content-Type: application/json" \
  -d '
 {
  "identity_document_id": "676a053b16781e43db7e75dcb1444ef8"
 }'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| identity_document_id | required | string | document id |

`required* = conditionally required`

```
 {
  "code": 0,
  "message": "Successful Response",
  "technical_message": "Successful Response",
  "details": [
    {
      "document": {
        "file_name": "430d595c22dbae26ef39ed91c3aabb49.jpg",
        "base64_content": "{base_64_endcoded_content}"
      }
    }
  ]
 }
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| code | number | genesis success code - 0 |
| message | string | const 'Successful Response' |
| technical_message | string | const 'Successful Response' |
| **details** | | |
| file_name | | |
| base64_content | | |

# Make call

This method is used to make a call or send an SMS to a given phone number. This method is used to complement the verification process. The system will make a call and dictates the verification code to be typed in the website. The following is a transcript of the voice message the system will use when en-US is used as language:

> Hello, thank you for using our phone verification system. Your code is [CODE]. Once again, your code is [CODE]. Goodbye!

The following is an example of the SMS text the system will use when en-US is used as language:

> Your code is [CODE]. Thank you.

`POST /kyc_service/create_authentication`

```
curl https://username:password@staging.gate.e-comprocessing.net/kyc_service/create_authentication \
  -X POST \
  -H "Content-Type: application/json" \
  -d '
 {
   "customer_username": "896342",
   "customer_unique_id": 2,
   "transaction_unique_id": "387783428790324",
   "customer_phone_number": "372489879342",
   "service_language": "bg",
   "security_code": "3423",
   "service_type": 1
 }'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| customer_username | optional | string | Username of the customer on your system |
| customer_unique_id | optional | string | Unique user identificator on your system |
| transaction_unique_id | required | string | Transaction identification in the merchants system; If not provided the system won't be able to link with the transaction that is being verified |
| customer_phone_number | required | string | Phone number to call; It must be complete country code + phone number; No dashes; For example: 50622560000 |
| service_language | required | string | a string value; See below |
| security_code | required | string | Numeric value - 4 digits only; It cannot start with 0; The boot is going to say this numeric value so the user can type it back on the website; |
| service_type | required | string | Numeric value to indicate if the system will send a text message or make a voice call; 1 for SMS; 2 for Voice call; |

`required* = conditionally required`

**Available languages for t_language**

| Language | Code |
|---|---|
| Arabic | a |
| Cantonese, Chinese/Hong Kong | zh-HK |
| Catalan | ca |
| Croatian | hr |
| Czech | cs |
| Danish | da |
| Dutch | nl |
| English, Australian | en-AU |
| English, UK | en-GB |
| English, US | en-US |
| Estonian | et |
| Filipino | fil |
| Finnish | fi |
| French | fr |
| French, Canadian | fr-CA |
| German | de |
| Greek | el |
| Hebrew | he |
| Hindi | hi |
| Hungarian | hu |
| Icelandic | is |
| Indonesian | id |
| Italian | it |
| Japanese | ja |

| Language | Code |
|---|---|
| Korean | ko |
| Latvian | lv |
| Lingala | ln |
| Lithuanian | lt |
| Mandarin | zh-CN |
| Norwegian | no |
| Polish | pl |
| Portuguese, Brazilian | pt-BR |
| Portuguese, European | pt |
| Romanian | ro |
| Russian | ru |
| Slovakian | sk |
| Spanish, European | es |
| Spanish, Latin American | es-419 |
| Swedish | sv |
| Thai | th |
| Turkish | tr |
| Ukrainian | uk |
| Vietnamese | vi |

```
{
  "code": 0,
  "message": "Successful Response",
  "technical_message": "Successful Response",
  "details": [
    {
      "reference_id": "233"
    }
  ]
}
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| code | number | genesis success code - 0 |
| message | string | const 'Successful Response' |
| technical_message | string | const 'Successful Response' |
| **details** | | |
| reference_id | | |

---

# Update call

This method is used to update the call status with the latest info received from the main system. It also updates the transaction associated with this verification call.

`POST /kyc_service/update_authentication`

```
curl https://username:password@staging.gate.e-comprocessing.net/kyc_service/update_authentication \
  -X POST \
  -H "Content-Type: application/json" \
  -d '
 {
  "reference_id": 1237834,
  "security_code_input": "4322",
  "verification_status": "4"
 }'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| reference_id | required | string | Unique value to identify the call in back office. |
| security_code_input | required | string | Transaction identification in the merchants system; If not provided the system won't be able to link with the transaction that is being verified |
| verification_status | required | string | The first two values are defined by the system when the call is created; the ones accepted in this call are the status 3; 4 and 5 only 1-In Progress; 2-Failed; 3-Verification Failed; 4-Verification Successful; 5-Abandon; |

`required* = conditionally required`

```
{
  "code": 0,
  "message": "Successful Response",
  "technical_message": "Successful Response",
  "details": [
    {
      "reference_id": "233"
    }
  ]
}
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| code | number | genesis success code - 0 |
| message | string | const 'Successful Response' |
| technical_message | string | const 'Successful Response' |
| **details** | | |
| reference_id | | |

---

# Error Response

```
{
  "code": 404,
  "message": "Passed attribute is invalid!",
  "technical_message": "The property '#/' did not contain a required property of 'customer_information'"
}
```

```
{
  "code": 803,
  "message": "KYC Services not configured!",
  "technical_message": "KYC Services not configured for Merchant!"
}
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| code | number | Genesis internal error code |
| message | string | Short explanation of occurred error |
| technical_message | string | More detailed explanation of occurred error |

---

# Kyc Service Notification

Since identity document verification is time consuming(can take up to 3-7 minutes) it is an asynchronous call. When verification is done a notification will be sent to configured Merchant notification url with all the details of the review along with the risk score. Note that multiple notifications can be expected so please always check the Score Complete parameter as if we have multiple DocumentId Verify providers, the score will be complemented with more info as we receive the data from the KYC sources. The count of expected notifications will be declared in initial IdentityDocumentUpload response.

KYC Notification Example

```
?kyc_source=OCR
&reference_id=1912
&score=10
&score_complete=0
&uid=5486354658
&error_message=
&analysis_ref_uid=148871
&controls_identifier=MODEL_VALIDITY
&controls_title_msg=OK
&controls_result_msg=OK
&controls_result=OK
&controls_control_identifier=MODEL_RECOGNIZED
&controls_control_title_msg=OK
&controls_control_result_msg=OK
&controls_control_result=OK
&document_classification_id_type=V
&document_detail_emit_country=USA
&document_detail_expiration_date_day=23
&document_detail_expiration_date_month=12
&document_detail_expiration_date_year=2004
&document_detail_document_number=555123ABC
&document_detail_extra_infos_data_key=PERSONAL_NUMBER
&document_detail_extra_infos_data_value=IFLND00AMS803085
&document_detail_extra_infos_title=Personal_Number
&holder_detail_last_name=TRAVELER
&holder_detail_first_name=HAPPYPERSON
&holder_detail_nationality=GBR
&holder_detail_gender=F
&holder_detail_birth_date_day=5
&holder_detail_birth_date_month=2
&holder_detail_birth_date_year=1965
&mrz_line1=VIUSATRAVELER_HAPPYPERSON
&mrz_line2=555123ABC6GBR6502056F0412236IFLND00AMS803085
&check_report_summary_check_identifier=SUMMARY_ID_COPY
&check_report_summary_check_title_msg=OK
&check_report_summary_check_result_msg=Original_Document
&check_report_summary_check_result=OK
&notification_type=kyc_service_execution
&signature=secure-signature
```

**Notification Parameters Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| kyc_source | string | name of the KYC source performed the validation |
| reference_id | string | Unique id for reference to document upload request |
| risk_score | number | Score product of validation rules |
| external_unique_id | string | Unique ID Reference from External Service Provider |
| analysis_reference_id | string | UniqueID Reference from Genesis |
| controls_identifier | string | List of all MAIN controls performed on the document |
| ... | string | Additional information related to verification process |
| notification_type | string | constant value ”kyc service response” |
| signature | string | the signature of the notification, should be used to verify the the notification was sent by Genesis |

The signature is a security measure meant to ensure that the gateway is really the sender of the notification. It is generated by concatenating the reference id with your API password and generating a SHA-512 Hash (Hex) of the string:

`SHA-512 Hash Hex of [reference_id][Your Merchant API password]`

**Notification signature examples**

| reference_id | API password | signature |
|---|---|---|
| 1912 | password1 | ecc5f441366aef60627a7fd2ad31e8c0e6d61d7cc3922e00fcdcf5cfbc2cc5ed67b9a3fa91a2c45ebd37c37acac6c543bed1f1146aea408c22741ffcd137fe75 |
| 1818 | password2 | e03a2cdda2d75b7522a1c0b14d078ef017b000aa419c872a1e6489b8dd287e6bc6573feece5cf0db9bd3e6711eb9a7cf65a29b7a13ba5bad1de13d16ba448283 |

When receiving the notification, you are required to render an XML page containing the transaction’s reference id so that the gateway knows that you have accepted the notification. If the XML is not delivered, the notification is sent periodically as per the rules for notifications delivery.

```
<?xml version="1.0" encoding="UTF-8"?>
<notification_echo>
  <notification_echo>reference_id</notification_echo>
</notification_echo>
```

# Genesis Fx Services

# General Info

Genesis Fx(Forex) Services provides the ability to retrieve up-to-date Fx rates. The API is synchronous and is based on RESTFul practices. Be sure to set Content-type: application/json in your headers.

To interact with the Fx API, you need to provide login credentials using standard HTTP Basic Authentication. (credentials can be found in your Admin interface.)

# Get Tiers

This call is used to return all Tiers that are related to your account.

`GET /v1/fx/tiers`

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://username:password@staging.gate.e-comprocessing.net/v1/fx/tiers \
  -X GET \
```

Successful Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
[
  {
    "id": 1,
    "name": "Tier Name",
    "description": "Tier Description",
    "tier_id": "Tier Identifier",
    "enabled": true
  }
]
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| id | number | tier id - 1 |
| name | string | name of the tier |
| description | string | description of the tier |
| tier_id | string | identification of the tier |
| enabled | boolean | state of the tier |

# Get Tier

This call is used to return information about selected Tier for your merchant.

`GET /v1/fx/tiers/:id`

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://username:password@staging.gate.e-comprocessing.net/v1/fx/tiers/1 \
  -X GET \
```

Successful Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
{
  "id": 1,
  "name": "Tier Name",
  "description": "Tier Description",
  "tier_id": "Tier Identifier",
  "enabled": true
}
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| id | number | tier id - 1 |
| name | string | name of the tier |
| description | string | description of the tier |
| tier_id | string | identification of the tier |
| enabled | boolean | state of the tier |

---

# Get Rates

This call is used to return all rates for Tier.

`GET /v1/fx/tiers/:tier_id/rates`

Note: `:tier_id` is the ID of Tier, not to be mistaken with `:tier_id` of the same entity.

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://username:password@staging.gate.e-comprocessing.net/v1/fx/tiers/1/rates \
  -X GET \
```

Successful Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
[
  {
    "id": 1,
    "source_currency": "Tier Name",
    "target_currency": "Tier Description",
    "trading_rate": "Tier Identifier",
    "enabled": null
  }
]
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| id | number | rate id - 1 |
| source_currency | string | source currency of the rate |
| target_currency | string | target currency of the rate |
| trading_rate | string | trading rate |

---

# Get Rate

This call is used to return information about selected Rate for merchant.`GET /v1/fx/tiers/:tier_id/rates/:id`

Note: `:tier_id` is the ID of Tier, not to be mistaken with `:tier_id` of the same entity.

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://username:password@staging.gate.e-comprocessing.net/v1/fx/tiers/1/rates/1 \
  -X GET \
```

Successful Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
{
  "id": 1,
  "source_currency": "Tier Name",
  "target_currency": "Tier Description",
  "trading_rate": "Tier Identifier",
  "enabled": null
}
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| id | number | rate id - 1 |
| source_currency | string | source currency of the rate |
| target_currency | string | target currency of the rate |
| trading_rate | string | trading rate |

---

# Search Rate

This call is used to return information about selected Rate by currency pair.

`POST /v1/fx/tiers/:tier_id/rates/search`

Note: `:tier_id` is the ID of Tier, not to be mistaken with `:tier_id` of the same entity.

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://username:password@staging.gate.e-comprocessing.net/v1/fx/tiers/:tier_id/rates/search \
  -X POST \
  -H "Content-Type: application/json" \
  -d '
 {
   "source_currency": "EUR",
   "target_currency": "USD"
 }'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| source_currency | required | string | source currency |
| target_currency | required | string | target currency |

`required* = conditionally required`

Successful Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
 {
   "id": 1,
   "source_currency": "Tier Name",
   "target_currency": "Tier Description",
   "trading_rate": "Tier Identifier",
   "enabled": null
 }
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| id | number | rate id - 1 |
| source_currency | string | source currency of the rate |
| target_currency | string | target currency of the rate |
| trading_rate | string | trading rate |

# Consumers

## Introduction

The Consumer entity brings Tokenization, Transactions and Web Payment Forms (WPF) together. It is a representation of a customer that can serve different purposes. A consumer is identified by providing both consumer ID and email. It is *explicitly* created via our Consumer API or *implicitly* by providing `customer_email` in either Transactions or WPF APIs. The main purpose of consumers is to group web payment forms and payment transactions. Using the merchant console, one can track consumers and find high-volume ones. The other role of consumers is to provide simplified, one-step tokenization of cardholder details. For Processing API that means securely storing card data in exchange for a token, which can be used for future payments. For WPF, customers can choose either to use previously stored cards or remember a new payment method.

---

## Consumer API

**CREATE CONSUMER**

Creates a consumer based on email address. Optionally, one can provide billing and shipping address. Addresses will be used, if none given, in Processing or WPF APIs.

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://staging.gate.e-comprocessing.net/v1/create_consumer/ \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <create_consumer_request>
   <email>consumer@email.com</email>
   <billing_address>
     <first_name>Travis</first_name>
     <last_name>Pastrana</last_name>
     <address1>Muster Str. 12</address1>
     <zip_code>10178</zip_code>
     <city>Los Angeles</city>
     <state>CA</state>
     <country>US</country>
   </billing_address>
   <shipping_address>
     <first_name>Travis</first_name>
     <last_name>Pastrana</last_name>
     <address1>Muster Str. 12</address1>
     <zip_code>10001</zip_code>
     <city>Los Angeles</city>
     <state>CA</state>
     <country>US</country>
   </shipping_address>
 </create_consumer_request>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| email | required | email address | The consumer email address must be unique. If another consumer exists with this email address, the request will be rejected. |
| **billing_address** | optional | | See Required vs Optional API params for details |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

Successful Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<create_consumer_response>
  <consumer_id>123456</consumer_id>
  <email>consumer@email.com</email>
  <status>enabled</status>
</create_consumer_response>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| consumer_id | string(10) | Consumer unique reference |
| email | email address | Consumer email address |
| status | string | Status of the consumer |

Error Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<create_consumer_response>
  <status>error</status>
  <code>330</code>
  <technical_message>Invalid email format!</technical_message>
  <message>Something went wrong, please contact support!</message>
</create_consumer_response>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| status | string | Status of the consumer |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |

#### RETRIEVE CONSUMER

Retrieves consumer details based on consumer id or email.

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://staging.gate.e-comprocessing.net/v1/retrieve_consumer/ \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
<?xml version="1.0" encoding="UTF-8"?>
<retrieve_consumer_request>
  <consumer_id>123456</consumer_id>
</retrieve_consumer_request>'
```

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://staging.gate.e-comprocessing.net/v1/retrieve_consumer/ \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <retrieve_consumer_request>
   <email>consumer@email.com</email>
 </retrieve_consumer_request>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| consumer_id | required | string(10) | Consumer unique reference. Required if the email is not provided |
| email | required | email address | Consumer email address. Required if the email is not provided |

`required* = conditionally required`

Successful Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<retrieve_consumer_response>
  <consumer_id>123456</consumer_id>
  <email>consumer@email.com</email>
  <status>enabled</status>
</retrieve_consumer_response>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| consumer_id | string(10) | Consumer unique reference |
| email | email address | Consumer email address |
| status | string | Status of the consumer |

Error Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<retrieve_consumer_response>
  <status>error</status>
  <code>702</code>
  <technical_message>Consumer not found!</technical_message>
  <message>Something went wrong, please contact support!</message>
</retrieve_consumer_response>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| status | string | Status of the consumer |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |

#### UPDATE CONSUMER

Updates consumer email and addresses.

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://staging.gate.e-comprocessing.net/v1/update_consumer/ \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <update_consumer_request>
   <consumer_id>123456</consumer_id>
   <email>consumer@email.com</email>
   <billing_address>
     <first_name>Travis</first_name>
     <last_name>Pastrana</last_name>
     <address1>Muster Str. 12</address1>
     <zip_code>10178</zip_code>
     <city>Los Angeles</city>
     <state>CA</state>
     <country>US</country>
   </billing_address>
   <shipping_address>
     <first_name>Travis</first_name>
     <last_name>Pastrana</last_name>
     <address1>Muster Str. 12</address1>
     <zip_code>10001</zip_code>
     <city>Los Angeles</city>
     <state>CA</state>
     <country>US</country>
   </shipping_address>
 </update_consumer_request>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| consumer_id | required | string(10) | Consumer unique reference |
| email | required | email address | New email address |
| **billing_address** | optional | | See Required vs Optional API params for details |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |
| **shipping_address** | optional | | |
| first_name | optional | string(255) | Customer first name |
| last_name | optional | string(255) | Customer last name |
| address1 | optional | string(255) | Primary address |
| address2 | optional | string(255) | Secondary address |
| zip_code | optional | string | ZIP code |
| city | optional | string(255) | City |
| state | optional | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | optional | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

Successful Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<update_consumer_response>
  <consumer_id>123456</consumer_id>
  <email>consumer@email.com</email>
  <status>enabled</status>
</update_consumer_response>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| consumer_id | string(10) | Consumer unique reference |
| email | email address | Consumer email address |
| status | string | Status of the consumer |

Error Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<update_consumer_response>
  <status>error</status>
  <code>702</code>
  <technical_message>Consumer not found!</technical_message>
  <message>Something went wrong, please contact support!</message>
</update_consumer_response>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| status | string | Status of the consumer |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |

#### DISABLE CONSUMER

Disable consumer from usage until further action.

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://staging.gate.e-comprocessing.net/v1/disable_consumer/ \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
<?xml version="1.0" encoding="UTF-8"?>
<disable_consumer_request>
  <consumer_id>123456</consumer_id>
  <email>consumer@email.com</email>
</disable_consumer_request>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| consumer_id | required | string(10) | Consumer unique reference |
| email | required | email address | Consumer email address |

`required* = conditionally required`

Successful Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<disable_consumer_response>
  <consumer_id>123456</consumer_id>
  <email>consumer@email.com</email>
  <status>disabled</status>
</disable_consumer_response>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| consumer_id | string(10) | Consumer unique reference |
| email | email address | Consumer email address |
| status | string | Status of the consumer |

Error Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<disable_consumer_response>
  <status>error</status>
  <code>702</code>
  <technical_message>Consumer not found!</technical_message>
  <message>Something went wrong, please contact support!</message>
</disable_consumer_response>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| status | string | Status of the consumer |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |

#### ENABLE CONSUMER

Enable consumer that was disabled in the past.

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://staging.gate.e-comprocessing.net/v1/enable_consumer/ \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <enable_consumer_request>
   <consumer_id>123456</consumer_id>
   <email>consumer@email.com</email>
 </enable_consumer_request>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| consumer_id | required | string(10) | Consumer unique reference |
| email | required | email address | Consumer email address |

`required* = conditionally required`

Successful Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
 <?xml version="1.0" encoding="UTF-8"?>
 <enable_consumer_response>
   <consumer_id>123456</consumer_id>
   <email>consumer@email.com</email>
   <status>enabled</status>
 </enable_consumer_response>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| consumer_id | string(10) | Consumer unique reference |
| email | email address | Consumer email address |
| status | string | Status of the consumer |

Error Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
 <?xml version="1.0" encoding="UTF-8"?>
 <enable_consumer_response>
   <status>error</status>
   <code>702</code>
   <technical_message>Consumer not found!</technical_message>
   <message>Something went wrong, please contact support!</message>
 </enable_consumer_response>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| status | string | Status of the consumer |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |

#### GET CONSUMER CARDS

Get previously tokenized card details for a consumer.

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://staging.gate.e-comprocessing.net/v1/get_consumer_cards/ \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <get_consumer_cards_request>
   <consumer_id>123456</consumer_id>
   <email>consumer@email.com</email>
 </get_consumer_cards_request>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| consumer_id | required | string(10) | Consumer unique reference |
| email | required | email address | Consumer email address |

`required* = conditionally required`

Successful Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
 <?xml version="1.0" encoding="UTF-8"?>
 <get_consumer_cards_response>
   <total>1</total>
   <card>
     <card_number>409603...0106</card_number>
     <card_holder>Travis Pastrana</card_holder>
     <expiration_month>12</expiration_month>
     <expiration_year>2021</expiration_year>
     <card_brand>master</card_brand>
   </card>
 </get_consumer_cards_response>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| total | string(255) | Number of non-expired consumer cards |
| **card** | | |
| card_number | | |
| card_holder | | |
| expiration_month | | |
| expiration_year | | |
| card_brand | | |

Error Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<get_consumer_cards_response>
  <status>error</status>
  <code>702</code>
  <technical_message>Consumer not found!</technical_message>
  <message>Something went wrong, please contact support!</message>
</get_consumer_cards_response>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| status | string | Status of the consumer |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |

# Tokenization

## Introduction

Tokenization is the process of replacing sensitive cardholder data with a surrogate value ("token"). The data to be tokenized must include at least the primary account number (PAN).

Tokenization greatly reduces the sensitive data that businesses need to store, thus improving security of credit card transactions and minimizing the costs related to PCI DSS compliance.

We issue reversible non-cryptographic tokens to merchants via our Tokenization API and take care to store safely the tokenized cardholder data. Merchants are able to use the issued tokens instead of the cardholder data when creating credit card transactions via our Processing API. PCI DSS compliant merchants have also the possibility to exchange the token for the original cardholder data via our Tokenization API ("detokenization").

## Tokenization API

**ACCEPTED CARDHOLDER PARAMETERS**

All cardholder data parameters are accepted for tokenization - card number, cardholder, expiration year, expiration month. Please note - CVV is not accepted.

**CONSUMER REQUIRED**

An enabled consumer is required in order to use this API. You have to create one or use existing, please check Consumer API.

**TOKENIZE**

Tokenizes cardholder data and issues a corresponding token. Merchants should take care to save the plain-text token value in their system as once issued it is not possible to obtain it again. Attempting to tokenize the same cardholder data will issue a new token. The token can be used in the Processing API instead of the cardholder data.

> Please note, CVV can't be part of the cardholder data.

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://staging.gate.e-comprocessing.net/v1/tokenize/ \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
<?xml version="1.0" encoding="UTF-8"?>
<tokenize_request>
  <consumer_id>123456</consumer_id>
  <email>consumer@email.com</email>
  <token_type>uuid</token_type>
  <card_data>
    <card_number>4200000000000000</card_number>
    <card_holder>John Doe</card_holder>
    <expiration_month>05</expiration_month>
    <expiration_year>2021</expiration_year>
  </card_data>
</tokenize_request>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| consumer_id | required | string(10) | Consumer unique reference |
| email | required | e-mail address | Consumer e-mail address |
| token_type | required | uuid | Token type format |
| **card_data** | required | | |
| card_number | required | string(13..21) | Complete cc number of customer |
| card_holder | optional | string(255) | Full name of customer as printed on credit card (first name and last name at least) |
| expiration_month | optional | MM | Expiration month as printed on credit card |
| expiration_year | optional | YYYY | Expiration year as printed on credit card |

`required* = conditionally required`

Successful Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<tokenize_response>
  <status>active</status>
  <token_id>34567</token_id>
  <token>ee946db8-d7db-4bb7-b608-b65b153e127d</token>
  <token_type>uuid</token_type>
  <consumer_id>123456</consumer_id>
</tokenize_response>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| status | string | Status of the token |
| token_id | string(32) | Unique token id |
| token | string(36) | Plain-text token value |
| token_type | uuid | Token type format |
| consumer_id | string(10) | Consumer unique reference |

Error Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<tokenize_response>
  <status>error</status>
  <code>720</code>
  <technical_message>Invalid token type!</technical_message>
  <message>Something went wrong, please contact support!</message>
</tokenize_response>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| status | string | Status of the token |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |

| Parameter | Type | Description |
|---|---|---|
| message | string(255) | Human readable error message which can be displayed to users. |

#### DETOKENIZE

Exchanges the token with the tokenized cardholder data

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://staging.gate.e-comprocessing.net/v1/detokenize/ \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <detokenize_request>
   <consumer_id>123456</consumer_id>
   <email>consumer@email.com</email>
   <token>ee946db8-d7db-4bb7-b608-b65b153e127d</token>
   <token_type>uuid</token_type>
 </detokenize_request>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| consumer_id | required | string(10) | Consumer unique reference |
| email | required | e-mail address | Consumer e-mail address |
| token | required | string(36) | Plain-text token value |
| token_type | required | uuid | Token type format |

`required* = conditionally required`

Successful Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<detokenize_response>
  <status>active</status>
  <token_id>34567</token_id>
  <token_type>uuid</token_type>
  <card_data>
    <card_number>4200000000000000</card_number>
    <card_holder>John Doe</card_holder>
    <expiration_month>05</expiration_month>
    <expiration_year>2021</expiration_year>
  </card_data>
</detokenize_response>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| status | string | Status of the token |
| token_id | string(32) | Unique token id |
| token | string(36) | Plain-text token value |
| token_type | uuid | Token type format |
| **card_data** | | |
| card_number | | |
| card_holder | | |

| Parameter | Type | Description |
|---|---|---|
| expiration_month | | |
| expiration_year | | |

Error Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<detokenize_response>
  <status>error</status>
  <code>730</code>
  <technical_message>Invalid token!</technical_message>
  <message>Something went wrong, please contact support!</message>
</detokenize_response>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| status | string | Status of the token |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |

#### UPDATE TOKEN

Updates the tokenized data corresponding to an existing valid token.

> Please note, PAN can't be updated

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://staging.gate.e-comprocessing.net/v1/update_token/ \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <update_token_request>
    <consumer_id>123456</consumer_id>
    <email>consumer@email.com</email>
    <token>ee946db8-d7db-4bb7-b608-b65b153e127d</token>
    <token_type>uuid</token_type>
    <card_data>
      <card_number>4200000000000000</card_number>
      <card_holder>John Doe</card_holder>
      <expiration_month>05</expiration_month>
      <expiration_year>2021</expiration_year>
    </card_data>
  </update_token_request>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| consumer_id | required | string(10) | Consumer unique reference |
| email | required | e-mail address | Consumer e-mail address |
| token | required | string(36) | Plain-text token value |
| token_type | required | uuid | Token type format |

| Parameter | Required | Format | Description |
|---|---|---|---|
| **card_data** | required | | |
| card_number | required | string(13..21) | Complete cc number of customer |
| card_holder | optional | string(255) | Full name of customer as printed on credit card (first name and last name at least) |
| expiration_month | optional | MM | Expiration month as printed on credit card |
| expiration_year | optional | YYYY | Expiration year as printed on credit card |

`required* = conditionally required`

Successful Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<update_token_response>
  <status>active</status>
  <token>ee946db8-d7db-4bb7-b608-b65b153e127d</token>
  <token_type>uuid</token_type>
</update_token_response>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| status | string | Status of the token |
| token | string(36) | Plain-text token value |
| token_type | uuid | Token type format |

Error Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<update_token_response>
  <status>error</status>
  <code>730</code>
  <technical_message>Invalid token!</technical_message>
  <update_token_response>message'</update_token_response>
</update_token_response>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| status | string | Status of the token |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |

#### VALIDATE TOKEN

Validates if a token is active, owned by a merchant, etc.

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://staging.gate.e-comprocessing.net/v1/validate_token/ \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <validate_token_request>
    <consumer_id>123456</consumer_id>
    <email>consumer@email.com</email>
    <token>ee946db8-d7db-4bb7-b608-b65b153e127d</token>
    <token_type>uuid</token_type>
  </validate_token_request>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| consumer_id | required | string(10) | Consumer unique reference |
| email | required | e-mail address | Consumer e-mail address |
| token | required | string(36) | Plain-text token value |
| token_type | required | uuid | Token type format |

`required* = conditionally required`

Successful Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
  <?xml version="1.0" encoding="UTF-8"?>
  <validate_token_response>
    <status>active</status>
    <token_id>34567</token_id>
    <token_type>uuid</token_type>
  </validate_token_response>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| status | string | Status of the token |
| token_id | string(32) | Unique token id |
| token_type | uuid | Token type format |

Error Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
  <?xml version="1.0" encoding="UTF-8"?>
  <validate_token_response>
    <status>error</status>
    <code>720</code>
    <technical_message>Invalid token type!</technical_message>
    <message>Something went wrong, please contact support!</message>
  </validate_token_response>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| status | string | Status of the token |
| code | integer | Error code according to Error code table |

| Parameter | Type | Description |
|---|---|---|
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |

#### DELETE TOKEN

Deletes a token.

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://staging.gate.e-comprocessing.net/v1/delete_token/ \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <delete_token_request>
    <consumer_id>123456</consumer_id>
    <email>consumer@email.com</email>
    <token>ee946db8-d7db-4bb7-b608-b65b153e127d</token>
    <token_type>uuid</token_type>
  </delete_token_request>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| consumer_id | required | string(10) | Consumer unique reference |
| email | required | e-mail address | Consumer e-mail address |
| token | required | string(36) | Plain-text token value |
| token_type | required | uuid | Token type format |

`required* = conditionally required`

Successful Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
  <?xml version="1.0" encoding="UTF-8"?>
  <delete_token_response>
    <status>active</status>
    <token_id>34567</token_id>
    <token_type>uuid</token_type>
  </delete_token_response>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| status | string | Status of the token |
| token_id | string(32) | Unique token id |
| token_type | uuid | Token type format |

Error Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<delete_token_response>
  <status>error</status>
  <code>720</code>
  <technical_message>Invalid token type!</technical_message>
  <message>Something went wrong, please contact support!</message>
</delete_token_response>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| status | string | Status of the token |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |

#### GET CARD

Exchanges the token with the tokenized masked cardholder data

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://staging.gate.e-comprocessing.net/v1/get_card/ \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <get_card_request>
   <consumer_id>123456</consumer_id>
   <email>consumer@email.com</email>
   <token>ee946db8-d7db-4bb7-b608-b65b153e127d</token>
   <token_type>uuid</token_type>
 </get_card_request>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| consumer_id | required | string(10) | Consumer unique reference |
| email | required | e-mail address | Consumer e-mail address |
| token | required | string(36) | Plain-text token value |
| token_type | required | uuid | Token type format |

`required* = conditionally required`

Successful Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<get_card_response>
  <status>active</status>
  <token_id>34567</token_id>
  <token_type>uuid</token_type>
  <card_data>
    <card_number>420000...0000</card_number>
    <card_holder>John Doe</card_holder>
    <expiration_month>05</expiration_month>
    <expiration_year>2021</expiration_year>
    <card_brand>visa</card_brand>
  </card_data>
</get_card_response>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| status | string | Status of the token |
| token_id | string(32) | Unique token id |
| token | string(36) | Plain-text token value |
| token_type | uuid | Token type format |
| **card_data** | | |
| card_number | | |
| card_holder | | |
| expiration_month | | |
| expiration_year | | |

Error Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<detokenize_response>
  <status>error</status>
  <code>730</code>
  <technical_message>Invalid token!</technical_message>
  <message>Something went wrong, please contact support!</message>
</detokenize_response>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| status | string | Status of the token |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |

---

# How to tokenize cardholder data in Processing API

In order to tokenize card details you need to set the `remember_card` flag to "true".

#### CREATE A CONSUMER

Please provide `customer_email` in addition to cardholder details and the `remember_card` flag. This will create a consumer and tokenize cardholder details. `consumer_id` and `token` will be returned in the response and notification. `consumer_id` will be returned in the reconcile response.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Sale');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('4200000000000000')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')
        ->setRememberCard('true');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.SaleRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    SaleRequest request = new SaleRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("4200000000000000");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");
    request.setRememberCard("true");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.sale(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "4200000000000000",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    },
    "remember_card": "true"
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://staging.gate.e-comprocessing.net/process/TERMINAL_TOKEN/ \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>sale</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>100</amount>
    <currency>USD</currency>
    <card_holder>Travis Pastrana</card_holder>
    <card_number>4200000000000000</card_number>
    <expiration_month>12</expiration_month>
    <expiration_year>2021</expiration_year>
    <cvv>834</cvv>
    <customer_email>travis@example.com</customer_email>
    <customer_phone>+1987987987987</customer_phone>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>10178</zip_code>
      <city>Los Angeles</city>
      <state>CA</state>
      <country>US</country>
    </billing_address>
    <remember_card>true</remember_card>
  </payment_transaction>'
```

**USE EXISTING CONSUMER**

Please provide `consumer_id` and `customer_email` in addition to cardholder details and the `remember_card` flag. An existing consumer will be used, if identified. `consumer_id` and `token` will be returned in the response and notification. `consumer_id` will be returned in the reconcile response.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Sale');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('4200000000000000')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')
        ->setRememberCard('true')
        ->setConsumerId('123456');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.SaleRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    SaleRequest request = new SaleRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("4200000000000000");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");
    request.setRememberCard("true");
    request.setConsumerId("123456");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.sale(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "4200000000000000",
    "expiration_month": "12",
    "expiration_year": 2021,
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    },
    "remember_card": "true",
    "consumer_id": "123456"
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://staging.gate.e-comprocessing.net/process/TERMINAL_TOKEN/ \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>sale</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>100</amount>
    <currency>USD</currency>
    <card_holder>Travis Pastrana</card_holder>
    <card_number>4200000000000000</card_number>
    <expiration_month>12</expiration_month>
    <expiration_year>2021</expiration_year>
    <cvv>834</cvv>
    <customer_email>travis@example.com</customer_email>
    <customer_phone>+1987987987987</customer_phone>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>10178</zip_code>
      <city>Los Angeles</city>
      <state>CA</state>
      <country>US</country>
    </billing_address>
    <remember_card>true</remember_card>
    <consumer_id>123456</consumer_id>
  </payment_transaction>'
```

---

# How to use tokens in Processing API

**REQUESTS**

Merchants can substitute cardholder parameters with tokens in the request when creating credit card transactions via our Processing API. Please note: merchants may choose to tokenize all cardholder parameters or only a subset. In the latter case the remaining parameters would need to be provided in the request. An enabled consumer is required to use tokens in Processing API. Please provide `consumer_id` and `customer_email` along with the `token`.

All Cardholder Parameters Have Been Tokenized

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Sale');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setToken('ee946db8-d7db-4bb7-b608-b65b153e127d')
        ->setCvv('834')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')
        ->setConsumerId('123456');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.SaleRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    SaleRequest request = new SaleRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setToken("ee946db8-d7db-4bb7-b608-b65b153e127d");
    request.setCvv("834");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");
    request.setConsumerId("123456");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.sale(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "token": "ee946db8-d7db-4bb7-b608-b65b153e127d",
    "cvv": "834",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "consumer_id": "123456"
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://staging.gate.e-comprocessing.net/process/TERMINAL_TOKEN/ \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>sale</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>100</amount>
    <currency>USD</currency>
    <token>ee946db8-d7db-4bb7-b608-b65b153e127d</token>
    <cvv>834</cvv>
    <customer_email>travis@example.com</customer_email>
    <customer_phone>+1987987987987</customer_phone>
    <consumer_id>123456</consumer_id>
  </payment_transaction>'
```

Only Pan Has Been Tokenized

Request

```
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Sale');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setToken('ee946db8-d7db-4bb7-b608-b65b153e127d')
        ->setCardHolder('Travis Pastrana')
        ->setCvv('834')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')
        ->setConsumerId('123456');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.SaleRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    SaleRequest request = new SaleRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setToken("ee946db8-d7db-4bb7-b608-b65b153e127d");
    request.setCardHolder("Travis Pastrana");
    request.setCvv("834");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");
    request.setConsumerId("123456");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.sale(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "token": "ee946db8-d7db-4bb7-b608-b65b153e127d",
    "card_holder": "Travis Pastrana",
    "cvv": "834",
    "expiration_month": "12",
    "expiration_year": 2021,
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "consumer_id": "123456"
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://staging.gate.e-comprocessing.net/process/TERMINAL_TOKEN/ \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>sale</transaction_type>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>100</amount>
    <currency>USD</currency>
    <token>ee946db8-d7db-4bb7-b608-b65b153e127d</token>
    <card_holder>Travis Pastrana</card_holder>
    <cvv>834</cvv>
    <expiration_month>12</expiration_month>
    <expiration_year>2021</expiration_year>
    <customer_email>travis@example.com</customer_email>
    <customer_phone>+1987987987987</customer_phone>
    <consumer_id>123456</consumer_id>
  </payment_transaction>'
```

---

# How to tokenize cardholder data in WPF API

In order to offer saving a payment method for future use, set the `remember_card` flag to "true".

#### CREATE A CONSUMER

Please provide `customer_email` in addition to the `remember_card` flag. This will create a consumer and offer the user to save cardholder details (tokenize). `consumer_id` will be returned in the response and reconcile response. `consumer_id` and `token` will be returned in the notification.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('WPF\Create');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setDescription('You are about to buy 3 shoes at www.shoes.com!')
        ->setNotificationUrl('https://www.example.com/notification')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setReturnCancelUrl('http://www.example.com/cancel.html')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')
        ->setLifetime('60')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')


        // Risk Params
        ->setRiskUserId('123456')


        // Transaction Types
        ->addTransactionType('authorize')
        ->addTransactionType('sale')
        ->setRememberCard('true');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.wpf.WPFCreateRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;
import java.net.MalformedURLException;
import java.net.URL;

public class GenesisExample {
  public static void main() throws MalformedURLException {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    WPFCreateRequest request = new WPFCreateRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setDescription("You are about to buy 3 shoes at www.shoes.com!");
    request.setNotificationUrl(new URL("https://www.example.com/notification"));
    request.setReturnSuccessUrl(new URL("http://www.example.com/success"));
    request.setReturnFailureUrl(new URL("http://www.example.com/failure"));
    request.setReturnCancelUrl(new URL("http://www.example.com/cancel.html"));
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");
    request.setLifetime(60);


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");


    // Risk Params
    request.setRiskUserId("123456");


    // Transaction Types
    request.addTransactionType("authorize").done();
    request.addTransactionType("sale").done();
    request.setRememberCard("true");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.wpf_create(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "description": "You are about to buy 3 shoes at www.shoes.com!",
    "notification_url": "https://www.example.com/notification",
    "return_success_url": "http://www.example.com/success",
    "return_failure_url": "http://www.example.com/failure",
    "return_cancel_url": "http://www.example.com/cancel.html",
    "amount": "100",
    "currency": "USD",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "lifetime": 60,
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    },
    "risk_params": {
      "user_id": "123456"
    },
    "transaction_types": [
      "authorize",
      "sale"
    ],
    "remember_card": "true"
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://staging.wpf.e-comprocessing.net/wpf \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <wpf_payment>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <description>You are about to buy 3 shoes at www.shoes.com!</description>
    <notification_url>https://www.example.com/notification</notification_url>
    <return_success_url>http://www.example.com/success</return_success_url>
    <return_failure_url>http://www.example.com/failure</return_failure_url>
    <return_cancel_url>http://www.example.com/cancel.html</return_cancel_url>
    <amount>100</amount>
    <currency>USD</currency>
    <customer_email>travis@example.com</customer_email>
    <customer_phone>+1987987987987</customer_phone>
    <lifetime>60</lifetime>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>10178</zip_code>
      <city>Los Angeles</city>
      <state>CA</state>
      <country>US</country>
    </billing_address>
    <risk_params>
      <user_id>123456</user_id>
    </risk_params>
    <transaction_types>
      <transaction_type>authorize</transaction_type>
      <transaction_type>sale</transaction_type>
    </transaction_types>
    <remember_card>true</remember_card>
  </wpf_payment>'
```

**USE EXISTING CONSUMER**

Please provide `consumer_id` and `customer_email` in addition to the `remember_card` flag. An existing consumer will be used, if identified, and offer the user to save cardholder details (tokenize). `consumer_id` will be returned in the response and reconcile response. `consumer_id` and `token` will be returned in the notification.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('WPF\Create');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setDescription('You are about to buy 3 shoes at www.shoes.com!')
        ->setNotificationUrl('https://www.example.com/notification')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setReturnCancelUrl('http://www.example.com/cancel.html')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')
        ->setLifetime('60')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')


        // Risk Params
        ->setRiskUserId('123456')


        // Transaction Types
        ->addTransactionType('authorize')
        ->addTransactionType('sale')
        ->setRememberCard('true')
        ->setConsumerId('123456');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.wpf.WPFCreateRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;
import java.net.MalformedURLException;
import java.net.URL;

public class GenesisExample {
  public static void main() throws MalformedURLException {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    WPFCreateRequest request = new WPFCreateRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setDescription("You are about to buy 3 shoes at www.shoes.com!");
    request.setNotificationUrl(new URL("https://www.example.com/notification"));
    request.setReturnSuccessUrl(new URL("http://www.example.com/success"));
    request.setReturnFailureUrl(new URL("http://www.example.com/failure"));
    request.setReturnCancelUrl(new URL("http://www.example.com/cancel.html"));
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");
    request.setLifetime(60);


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");


    // Risk Params
    request.setRiskUserId("123456");


    // Transaction Types
    request.addTransactionType("authorize").done();
    request.addTransactionType("sale").done();
    request.setRememberCard("true");
    request.setConsumerId("123456");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.wpf_create(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "description": "You are about to buy 3 shoes at www.shoes.com!",
    "notification_url": "https://www.example.com/notification",
    "return_success_url": "http://www.example.com/success",
    "return_failure_url": "http://www.example.com/failure",
    "return_cancel_url": "http://www.example.com/cancel.html",
    "amount": "100",
    "currency": "USD",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "lifetime": 60,
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    },
    "risk_params": {
      "user_id": "123456"
    },
    "transaction_types": [
      "authorize",
      "sale"
    ],
    "remember_card": "true",
    "consumer_id": "123456"
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://staging.wpf.e-comprocessing.net/wpf \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <wpf_payment>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <description>You are about to buy 3 shoes at www.shoes.com!</description>
    <notification_url>https://www.example.com/notification</notification_url>
    <return_success_url>http://www.example.com/success</return_success_url>
    <return_failure_url>http://www.example.com/failure</return_failure_url>
    <return_cancel_url>http://www.example.com/cancel.html</return_cancel_url>
    <amount>100</amount>
    <currency>USD</currency>
    <customer_email>travis@example.com</customer_email>
    <customer_phone>+1987987987987</customer_phone>
    <lifetime>60</lifetime>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>10178</zip_code>
      <city>Los Angeles</city>
      <state>CA</state>
      <country>US</country>
    </billing_address>
    <risk_params>
      <user_id>123456</user_id>
    </risk_params>
    <transaction_types>
      <transaction_type>authorize</transaction_type>
      <transaction_type>sale</transaction_type>
    </transaction_types>
    <remember_card>true</remember_card>
    <consumer_id>123456</consumer_id>
  </wpf_payment>'
```

---

## How to use tokens in WPF API

Please provide `consumer_id` and `customer_email`. An existing consumer will be used, if identified, and offer the user to select a previously stored card for payment.

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('WPF\Create');
    $request = $genesis->request();

    $request
        ->setTransactionId('43671')
        ->setUsage('40208 concert tickets')
        ->setDescription('You are about to buy 3 shoes at www.shoes.com!')
        ->setNotificationUrl('https://www.example.com/notification')
        ->setReturnSuccessUrl('http://www.example.com/success')
        ->setReturnFailureUrl('http://www.example.com/failure')
        ->setReturnCancelUrl('http://www.example.com/cancel.html')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCustomerEmail('travis@example.com')
        ->setCustomerPhone('+1987987987987')
        ->setLifetime('60')


        // Billing Address
        ->setBillingFirstName('Travis')
        ->setBillingLastName('Pastrana')
        ->setBillingAddress1('Muster Str. 12')
        ->setBillingZipCode('10178')
        ->setBillingCity('Los Angeles')
        ->setBillingState('CA')
        ->setBillingCountry('US')


        // Risk Params
        ->setRiskUserId('123456')


        // Transaction Types
        ->addTransactionType('authorize')
        ->addTransactionType('sale')
        ->setConsumerId('123456');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.wpf.WPFCreateRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;
import java.net.MalformedURLException;
import java.net.URL;

public class GenesisExample {
  public static void main() throws MalformedURLException {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    WPFCreateRequest request = new WPFCreateRequest();

    request.setTransactionId("43671");
    request.setUsage("40208 concert tickets");
    request.setDescription("You are about to buy 3 shoes at www.shoes.com!");
    request.setNotificationUrl(new URL("https://www.example.com/notification"));
    request.setReturnSuccessUrl(new URL("http://www.example.com/success"));
    request.setReturnFailureUrl(new URL("http://www.example.com/failure"));
    request.setReturnCancelUrl(new URL("http://www.example.com/cancel.html"));
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCustomerEmail("travis@example.com");
    request.setCustomerPhone("+1987987987987");
    request.setLifetime(60);


    // Billing Address
    request.setBillingFirstname("Travis");
    request.setBillingLastname("Pastrana");
    request.setBillingPrimaryAddress("Muster Str. 12");
    request.setBillingZipCode("10178");
    request.setBillingCity("Los Angeles");
    request.setBillingState("CA");
    request.setBillingCountry("US");


    // Risk Params
    request.setRiskUserId("123456");


    // Transaction Types
    request.addTransactionType("authorize").done();
    request.addTransactionType("sale").done();
    request.setConsumerId("123456");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.wpf_create(
  {
    "transaction_id": "43671",
    "usage": "40208 concert tickets",
    "description": "You are about to buy 3 shoes at www.shoes.com!",
    "notification_url": "https://www.example.com/notification",
    "return_success_url": "http://www.example.com/success",
    "return_failure_url": "http://www.example.com/failure",
    "return_cancel_url": "http://www.example.com/cancel.html",
    "amount": "100",
    "currency": "USD",
    "customer_email": "travis@example.com",
    "customer_phone": "+1987987987987",
    "lifetime": 60,
    "billing_address": {
      "first_name": "Travis",
      "last_name": "Pastrana",
      "address1": "Muster Str. 12",
      "zip_code": "10178",
      "city": "Los Angeles",
      "state": "CA",
      "country": "US"
    },
    "risk_params": {
      "user_id": "123456"
    },
    "transaction_types": [
      "authorize",
      "sale"
    ],
    "consumer_id": "123456"
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://staging.wpf.e-comprocessing.net/wpf \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <wpf_payment>
    <transaction_id>43671</transaction_id>
    <usage>40208 concert tickets</usage>
    <description>You are about to buy 3 shoes at www.shoes.com!</description>
    <notification_url>https://www.example.com/notification</notification_url>
    <return_success_url>http://www.example.com/success</return_success_url>
    <return_failure_url>http://www.example.com/failure</return_failure_url>
    <return_cancel_url>http://www.example.com/cancel.html</return_cancel_url>
    <amount>100</amount>
    <currency>USD</currency>
    <customer_email>travis@example.com</customer_email>
    <customer_phone>+1987987987987</customer_phone>
    <lifetime>60</lifetime>
    <billing_address>
      <first_name>Travis</first_name>
      <last_name>Pastrana</last_name>
      <address1>Muster Str. 12</address1>
      <zip_code>10178</zip_code>
      <city>Los Angeles</city>
      <state>CA</state>
      <country>US</country>
    </billing_address>
    <risk_params>
      <user_id>123456</user_id>
    </risk_params>
    <transaction_types>
      <transaction_type>authorize</transaction_type>
      <transaction_type>sale</transaction_type>
    </transaction_types>
    <consumer_id>123456</consumer_id>
  </wpf_payment>'
```

---

## Supported transaction types

All transaction types accepting cardholder data are supported -Account Verification, Authorize, Authorize3D, Sale, Sale3D, InitRecurringSale, InitRecurringSale3D, Payout, Intersolve, Fashioncheque.

# Importation of external tokens and card details

#### CSV FILE FORMAT

**CSV file headers:**

| Header | req? | Description |
|---|---|---|
| token | required | Plain-text token value (merchant/PSP token to migrate to us) |
| token_type | optional | Token type format |
| status | optional | Status of the token |
| card_number | required | Credit card number |
| card_holder | required | Full name of customer as printed on cc (first name and last name at least) |
| expiration_month | required | Expiration month as printed on credit card |
| expiration_year | required | Expiration year as printed on credit card |
| email | required | Customer email |
| first_name | optional | Customer first name used for billing address details |
| last_name | optional | Customer last name used for billing address details |
| address1 | optional | Customer primary address used for billing address details |
| address2 | optional | Customer secondary address used for billing address details |
| zip_code | optional | Zip code used for billing address details |
| city | optional | Customer city used for billing address details |
| state | optional | State code in ISO 3166-2, required for USA and Canada, used for billing address details |
| country | optional | Country code in ISO 3166 used for billing address details |

CSV file delimiter must be ',' (comma).

#### ENCRYPTION OF THE CSV FILE

The CSV contains sensitive data and therefore it must be encrypted. A public GPG key should be used for this purpose.

Command to encrypt the file. It will generate encrypted file ending with .gpg extension:

```
gpg --encrypt --recipient test@email.com test_token_file.csv
```

#### UPLOADING THE ENCRYPTED CSV FILE TO REMOTE SFTP SERVER

Once the CSV file is encrypted it must be uploaded to our remote SFTP server for processing and importing its content.

The remote SFTP connection has the following options which should be provided on demand (please, contacttech-support@e-comprocessing.com for more details):

**host** - host where the remote SFTP server is located (ex. 1.2.3.4)
**user** - user allowed to connect to the remote SFTP server (ex. TEST_USER)
**port** - port listening for SFTP connections (ex. 12345)
**directory** - directory to upload/download encrypted CSV files (ex. import_dir/)
**ssh_key** - SSH key to authenticate the relevant user

Commands to upload the encrypted CSV file to remote SFTP server:

```
sftp -i "path-to-the-ssh-key-file" -P 12345 TEST_USER@1.2.3.4:import_dir/
```

Once connected to the remote SFTP:

```
sftp> put test_tokens_file.csv.gpg
Uploading test_tokens_file.csv.gpg to /import_dir/test_tokens_file.csv.gpg
sftp> quit
```

#### DOWNLOADING A RESPONSE CSV FILE FROM THE REMOTE SFTP SERVER

Once the tokens are imported in Genesis together with the relevant card details, a response CSV file would be generated. It will contain mapping between the tokens and the consumer ids and emails associated with them. They would be needed for successful usage of the imported tokens.

Commands to retrieve the response CSV files from the remote SFTP server:

```
sftp -i "path-to-the-ssh-key-file" -P 12345 TEST_USER@1.2.3.4:import_dir/
```

Once connected to the remote SFTP:

```
sftp> get test_tokens_file_response.csv
Fetching /import_dir/test_tokens_file_response.csv to test_tokens_file_response.csv
sftp> quit
```

# Transaction API

## Transaction Card Expiry Date Update API

Each card-based transaction has an expiration date, which can be updated using the Transaction Card Expiry Date Update API.

The API endpoint is suitable for recurring payments where the card has been renewed and has now a different expiration date.

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://staging.gate.e-comprocessing.net/v1/transaction/expiry_date/:transaction_unique_id \
  -X PUT \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <update_card_expiration_request>
   <expiration_month>12</expiration_month>
   <expiration_year>2021</expiration_year>
 </update_card_expiration_request>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| expiration_month | required | MM | Expiration month as printed on credit card |
| expiration_year | required | YYYY | Expiration year as printed on credit card |

`required* = conditionally required`

> The provided expiration date must be in the future and after the current expiration date of the payment transaction

Successful Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<update_card_expiration_response>
  <status>success</status>
  <code>200</code>
  <message>Card expiry date updated successfully!</message>
</update_card_expiration_response>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| status | string(255) | Status of the response |
| code | integer | Successful code (200) |
| message | string(255) | Human readable error message which can be displayed to users. |

Error Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<update_card_expiration_response>
  <status>error</status>
  <code>300</code>
  <technical_message>The provided expiration date must be in the future and after the current one</technical_message>
  <message>Please check input data for errors!</message>
</update_card_expiration_response>
```

**Error Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| status | string(255) | Status of the response |
| code | integer | Error code according to Error code table |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |
| message | string(255) | Human readable error message which can be displayed to users. |

# APM Services

## Alipay Register Merchant

## Introduction

Alipay Register Merchant call allows to register merchants of online payments into Alipay system or to update the registration information of merchants.

To interact with the Alipay Register Merchant API, you need to provide login credentials using standard HTTP Basic Authentication. Be sure to set Content-type: application/json in your headers. Replace terminal_id in the request with the desired terminal ID.

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://username:password@staging.gate.e-comprocessing.net/v1/alipay/register_merchant/:terminal_token \
  -X POST \
  -H "Content-Type: application/json" \
  -d '
 {
   "merchant_type": "ENTERPRISE",
   "site_information": {
     "type": "APP",
     "url": "https://example.com",
     "name": "Example"
   },
   "contact_number": "1234567",
   "registration_number": "1234567",
   "representative_id": "1234567",
   "representative_name": "Your Name"
 }'
```

Successful Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
 {
   "success": "true"
 }
```

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://username:password@staging.gate.e-comprocessing.net/v1/alipay/register_merchant/:terminal_token \
  -X POST \
  -H "Content-Type: application/json" \
  -d '
 {
   "merchant_type": "INDIVIDUAL",
   "site_information": {
     "type": "WEB",
     "url": "https://example.com",
     "name": "Example"
   },
   "contact_number": "1234567",
   "registration_number": "1234567",
   "shareholder_id": "1234567",
   "shareholder_name": "Your Name"
 }'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| merchant_type | required | string | Merchant type, the value can be INDIVIDUAL for the sole proprietorship or ENTERPRISE. |
| site_information | required | object | This field is in JSON object. See site_infos for details. |
| shareholder_id | required* | string(128) | ID or passport number, or business registration number of the primary shareholder of the merchant. Specify this field only when the merchant type is ENTERPRISE. |
| shareholder_name | required* | string(64) | Legal name of the primary shareholder of the merchant. Specify this field only when the merchant type is ENTERPRISE. |
| registration_number | required* | string(128) | Business registration number specified on the business registration document. Note: This field is not required when the merchant type is INDIVIDUAL and no registration number exists. |
| representative_name | required* | string(64) | Full legal name of the business owner. Specify this field only when the merchant type is INDIVIDUAL. This field is optional if the merchant type is ENTERPRISE. |
| representative_id | required* | string(128) | ID or passport number of the business owner. Specify this field only when the merchant type is INDIVIDUAL. This field is optional if the merchant type is ENTERPRISE. |
| settlement_number | optional | string(64) | Settlement bank account number of the merchant. Use letters and numbers only. |
| contact_number | required | string(64) | Contact phone number of the merchant, numbers and special characters +-() only. |

| Parameter | Required | Format | Description |
|---|---|---|---|
| contact_email | optional | string(128) | Contact email address of the merchant. |
| customer_service_number | optional | string(64) | Customer service phone number of the merchant, numbers and special characters +-() only. |
| customer_service_email | optional | string(128) | Customer service email address of the merchant. |

`required* = conditionally required`

**ALIPAY REGISTER MERCHANT SITE INFO PARAMETERS**

---

## Site info

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| type | required | string | Site type. For website URL, the value of this parameter must be WEB. For app download URL, the value of this parameter must be APP. Use uppercase. |
| url | required | string(256) | Site URL. When site_type is WEB, pass the URL in this format: http/https + SLD + TLD. When site_type is APP, pass the APP download URL starting with http/https. |
| name | optional | string(512) | Site name. |

`required* = conditionally required`

---

## Klarna

---

## Introduction

Klarna Services provides the ability for merchants to release remaining authorization, resend invoice, update order items or to update order address of a transaction.

> Klarna services requests are handled exactly like transaction requests via XML and authentication is required. Note that Klarna services can be done via transaction_id

**RELEASE REMAINING AUTHORIZATION API**

The URL for Release Remaining Authorization API is:

Production:

`https://gate.e-comprocessing.net/klarna/release_remaining_authorization`

Staging (for integration):

`https://staging.gate.e-comprocessing.net/klarna/release_remaining_authorization`

> Release Remaining Authorization service is used for approved Klarna transaction with at least one approved partial Klarna capture transaction.
> With this call the remaining authorization amount will be released and Klarna captures will be forbidden for the authorize transaction.

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://staging.gate.e-comprocessing.net/klarna/release_remaining_authorization \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <release_remaining_authorization_request>
   <transaction_id>a1qf12e81eb23d0e00ffb85b1db7d152</transaction_id>
 </release_remaining_authorization_request>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_id | required | string(255) | Unique transaction id defined by merchant |

`required* = conditionally required`

Successful Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<release_remaining_authorization_request>
  <status>success</status>
  <technical_message>Transaction operation successful!</technical_message>
</release_remaining_authorization_request>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| status | string(255) | Status of the transaction, see states |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |

#### RESEND INVOICE API

The URL for Resend Invoice API is:

Production:

`https://gate.e-comprocessing.net/klarna/resend_invoice`

Staging (for integration):

`https://staging.gate.e-comprocessing.net/klarna/resend_invoice`

> Resend Invoice service is used only for approved not yet refunded Klarna capture transaction.
> With this call Klarna will resend the invoice of the captured transaction.

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://staging.gate.e-comprocessing.net/klarna/resend_invoice \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
<?xml version="1.0" encoding="UTF-8"?>
<resend_invoice_request>
  <transaction_id>a1qf12e81eb23d0e00ffb85b1db7d152</transaction_id>
</resend_invoice_request>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_id | required | string(255) | Unique transaction id defined by merchant |

`required* = conditionally required`

Successful Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<resend_invoice_request>
  <status>success</status>
  <technical_message>Transaction operation successful!</technical_message>
</resend_invoice_request>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| status | string(255) | Status of the transaction, see states |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |

#### UPDATE ORDER ITEMS API

The URL for Update Order Items API is:

Production:

`https://gate.e-comprocessing.net/klarna/update_order_items`

Staging (for integration):

`https://staging.gate.e-comprocessing.net/klarna/update_order_items`

> Update Order Items service is used only for approved but not yet captured Klarna transaction.
> With this call amount and associated items will be updated.

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://staging.gate.e-comprocessing.net/klarna/update_order_items \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <update_order_items_request>
   <transaction_id>a1qf12e81eb23d0e00ffb85b1db7d152</transaction_id>
   <amount>10000</amount>
   <items>
     <item>
       <type>physical</type>
       <reference>19-402-BG1</reference>
       <name>BatteryPowerPack</name>
       <quantity>1</quantity>
       <unit_price>5000</unit_price>
       <tax_rate>0</tax_rate>
       <total_amount>5000</total_amount>
       <total_discount_amount>0</total_discount_amount>
       <total_tax_amount>0</total_tax_amount>
     </item>
   </items>
 </update_order_items_request>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| amount | required | integer > 0 | Amount of transaction in minor currency unit, seeCurrency and Amount Handling for details |
| **items** | required | | List with items |
| item_type | required | string(255) | Order line type. Possible values: Supported item types |
| quantity | required | integer | Non-negative. The item quantity |
| unit_price | required | integer | Minor units. Includes tax, excludes discount(max value: 100000000) |
| total_amount | required | integer | Includes tax and discount. Must match (quantity unit price) - total discount amount divided by quantity (max value: 100000000) |

| Parameter | Required | Format | Description |
|---|---|---|---|
| reference | optional | string(255) | Article number, SKU or similar |
| name | optional | string(255) | Descriptive item name |
| tax_rate | optional | integer | Non-negative. In percent, two implicit decimals. I.e 2500 = 25.00 percent |
| total_discount_amount | optional | integer | Non-negative minor units. Includes tax |
| total_tax_amount | optional | integer | Must be within 1 of total amount - total_amount * 10000 / (10000 + tax rate). Negative when type is discount |
| image_url | optional | url | URL to an image that can be later embedded in communications between Klarna and the customer. (max 1024 characters) |
| product_url | optional | url | URL to an image that can be later embedded in communications between Klarna and the customer. (max 1024 characters) |
| quantity_unit | optional | string(8) | Unit used to describe the quantity, e.g. kg, pcs... If defined has to be 1-8 characters |
| **product_identifiers** | optional | | List with product identifiers |
| brand | optional | string(255) | The product's brand name as generally recognized by consumers. If no brand is available for a product, do not supply any value |
| category_path | optional | string(255) | The product's category path as used in the merchant's webshop. Include the full and most detailed category and separate the segments with ' > ' |
| global_trade_item_number | optional | string(255) | The product's Global Trade Item Number (GTIN). Common types of GTIN are EAN, ISBN or UPC. Exclude dashes and spaces, where possible |
| manufacturer_part_number | optional | string(255) | The product's Manufacturer Part Number (MPN), which - together with the brand - uniquely identifies a product. Only submit MPNs assigned by a manufacturer and use the most specific MPN possible |
| **merchant_data** | optional | | List with merchant data |
| marketplace_seller_info | optional | string(255) | Information for merchant marketplace |

`required* = conditionally required`

Successful Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<update_order_items_request>
  <status>success</status>
  <technical_message>Transaction operation successful!</technical_message>
</update_order_items_request>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| status | string(255) | Status of the transaction, see states |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |

#### UPDATE ORDER ADDRESS API

The URL for Update Order Address API is:

Production:

`https://gate.e-comprocessing.net/klarna/update_order_address`

Staging (for integration):

`https://staging.gate.e-comprocessing.net/klarna/update_order_address`

> Update Order Address service is used only for approved but not yet captured Klarna transaction.
> With this call billing and shipping addresses will be updated.

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://staging.gate.e-comprocessing.net/klarna/update_order_address \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <update_order_address_request>
    <transaction_id>a1qf12e81eb23d0e00ffb85b1db7d152</transaction_id>
    <billing_address>
      <first_name>Barney</first_name>
      <last_name>Rubble</last_name>
      <address1>14, Nerazdelni str</address1>
      <zip_code>1407</zip_code>
      <city>Berlin</city>
      <country>DE</country>
    </billing_address>
    <shipping_address>
      <first_name>Barney</first_name>
      <last_name>Rubble</last_name>
      <address1>14, Nerazdelni str</address1>
      <zip_code>1407</zip_code>
      <city>Berlin</city>
      <country>DE</country>
    </shipping_address>
  </update_order_address_request>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| transaction_id | required | string(255) | Unique transaction id defined by merchant |
| **billing_address** | required | | See Required vs Optional API params for details |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |
| **shipping_address** | required | | |
| first_name | required* | string(255) | Customer first name |
| last_name | required* | string(255) | Customer last name |
| address1 | required* | string(255) | Primary address |
| address2 | required* | string(255) | Secondary address |
| zip_code | required* | string | ZIP code |
| city | required* | string(255) | City |
| state | required* | string(2) | State code in ISO 3166-2, required for USA and Canada |
| country | required | string(2) | Country code in ISO 3166 |

`required* = conditionally required`

Successful Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
<?xml version="1.0" encoding="UTF-8"?>
<update_order_address_request>
  <status>success</status>
  <technical_message>Transaction operation successful!</technical_message>
</update_order_address_request>
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| status | string(255) | Status of the transaction, see states |
| technical_message | string(255) | Technical error message (for internal use only, not to be displayed to users). |

---

# Trustly Select Account

---

## Introduction

Trustly Select Account call allows the merchant's customers to select their Trustly account, which is later on used to complete a Trustly Bank Pay-out transaction.

In the response of this call, the merchant will receive a URL. This URL is used by the customer in order to select the desired account in one of the banks that Trustly provides. Once the customer has completed the process, they will be returned back to failure or success URL given on the request. After that, the merchant will receive a notification on the set-up merchant processing URL.

> To interact with the Trustly Select Account API, you need to provide login credentials using standard HTTP Basic Authentication. Be sure to set Content-type: application/json in your headers. Replace terminal_id in the request with the desired terminal ID.

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://username:password@staging.gate.e-comprocessing.net/v1/trustly/select_account/:terminal_token \
  -X POST \
  -H "Content-Type: application/json" \
  -d '
 {
   "country": "UK",
   "first_name": "Travis",
   "last_name": "Pastrana",
   "ip_address": "255.255.255.255",
   "mobile_phone": "+441509813888",
   "national_id": "8910103344",
   "birth_date": "1989-10-10",
   "success_url": "https://example.com/success",
   "failure_url": "https://example.com/failure",
   "user_id": "12345678",
   "unique_id": "gp634ec5e7dbe6ca3871974accb875cd",
   "locale": "en_US"
 }'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| country | required | string(2) | Country code in ISO 3166 |
| first_name | required | string(255) | Customer first name |
| ip_address | required | string(255) | IP address of the customer |
| email | required | e-mail address | Must contain valid e-mail of the customer |
| mobile_phone | required | string(32) | Must contain valid phone number of the customer |
| national_id | required | string(20) | National Identifier number of the customer |
| birth_date | required | yyyy-mm-dd | Must contain valid birth date of the customer |
| success_url | required | url | URL where the customer is sent to after successful authentication |

| Parameter | Required | Format | Description |
|---|---|---|---|
| failure_url | required | url | URL where the customer is sent to after failed authentication |
| user_id | required | string(255) | Unique user identifier defined by merchant |
| unique_id | required | string(255) | Unique identifier defined by merchant |
| locale | required | string(20) | Customer's localization preference in the format [language[_territory]]. Language is theISO 639 code and the territory ISO 3166 code. |

`required* = conditionally required`

Successful Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
{
  "order_id": "4014321",
  "url": "http://example.com/customer_redirect"
}
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| order_id | string(255) | Order identifier |
| url | url | URL where user has to be redirected to complete select account process. |

---

# Trustly Register Account

---

# Introduction

Trustly Register Account call allows the merchant to verify customer's bank account details and get the associated account id which can be saved and used for future payouts.

To interact with the Trustly Register Account API, you need to provide login credentials using standard HTTP Basic Authentication. Be sure to set Content-type: application/json in your headers. Replace terminal_id in the request with the desired terminal ID.

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://username:password@staging.gate.e-comprocessing.net/v1/trustly/register_account/:terminal_token \
  -X POST \
  -H "Content-Type: application/json" \
  -d '
 {
   "first_name": "Travis",
   "last_name": "Pastrana",
   "mobile_phone": "+441509813888",
   "national_id": "8910103344",
   "birth_date": "1989-10-10",
   "user_id": "12345678",
   "clearing_house": "SPAIN",
   "account_number": "ES8701820004756386447000",
   "bank_number": ""
 }'
```

**List of supported clearinghouses:**

AUSTRIA, BELGIUM, BULGARIA, CROATIA, CYPRUS, CZECH_REPUBLIC, DENMARK, ESTONIA, FINLAND, FRANCE, GERMANY, GREECE, HUNGARY, IRELAND, ITALY, LATVIA,

LITHUANIA, LUXEMBOURG, MALTA, NETHERLANDS, NORWAY, POLAND, PORTUGAL, ROMANIA, SLOVAKIA, SLOVENIA, SPAIN, SWEDEN, UNITED_KINGDOM

Successful Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
{
  "account_id": "1044806532",
  "clearing_house": "SPAIN",
  "bank": "BBVA",
  "descriptor": "******************447000"
}
```

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://username:password@staging.gate.e-comprocessing.net/v1/trustly/register_account/:terminal_token \
  -X POST \
  -H "Content-Type: application/json" \
  -d '
 {
  "first_name": "Travis",
  "last_name": "Pastrana",
  "mobile_phone": "+441509813888",
  "national_id": "123456789",
  "birth_date": "1989-10-10",
  "user_id": "12345678",
  "clearing_house": "SWEDEN",
  "account_number": "8257466",
  "bank_number": "5839"
 }'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| first_name | required | string(255) | Customer first name |
| last_name | required | string(255) | Customer last name |
| email | optional | e-mail address | Must contain valid e-mail of the customer |
| mobile_phone | optional | string(32) | Must contain valid phone number of the customer |
| national_id | optional | string(20) | National Identifier number of the customer |
| birth_date | required | yyyy-mm-dd | Must contain valid birth date of the customer |
| user_id | required | string(255) | Unique user identifier defined by merchant |
| clearing_house | required | string(255) | The clearing house of the customer's bank account. Typically the name of a country in uppercase letters. |
| account_number | required | string(32) | The account number of the customer's bank account. Can be either IBAN or country-specific format. |
| bank_number | required | string(32) | The bank number of the customer's account in the given clearing house. For bank accounts in IBAN format, just provide an empty string ("") |

`required* = conditionally required`

List of clearing houses with IBAN support: AUSTRIA, BELGIUM, BULGARIA, CROATIA, CYPRUS, CZECH_REPUBLIC, DENMARK, ESTONIA, FINLAND, FRANCE, GERMANY, GREECE, HUNGARY, IRELAND, ITALY, LATVIA, LITHUANIA, LUXEMBOURG, MALTA, NETHERLANDS, NORWAY, POLAND, ROMANIA, SLOVAKIA, SLOVENIA, SPAIN, UNITED_KINGDOM

List of clearing houses with account number and bank number support: CZECH_REPUBLIC, DENMARK, ESTONIA, HUNGARY, NORWAY, PORTUGAL, SWEDEN, UNITED_KINGDOM

Successful Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
{
  "account_id": "7653385737",
  "clearing_house": "SWEDEN",
  "bank": "Skandiabanken",
  "descriptor": "*******4057"
}
```

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| account_id | string(255) | Unique identifier associated with the account |
| clearing_house | string(255) | The clearing house of the account |
| bank | string(255) | Name of the bank for this account |
| descriptor | string(255) | A descriptor for this account |

> The account_id received in the response should be stored to process further payout requests.

---

# TransferTo

---

## Introduction

TransferTo API endpoint provides merchants with the ability to retrieve an up-to-date list of TransferTo Payers. Those are the institutions that provide the money to the consumers.

> TransferTo Retrieve Payers request needs authentication.

#### RETRIEVE PAYERS API

This request is used to retrieve up-to-date TransferTo Payers list.

`GET /transfer_to_payers/payers`

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://username:password@staging.gate.e-comprocessing.net/transfer_to_payers/payers \
  -X GET \
```

Successful Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<payers_response>
  <payers>
    <payer>
      <id>1</id>
      <name>Sample Payer 1</name>
      <currency>USD</currency>
      <country_iso_code>USA</country_iso_code>
      <service>BankAccount</service>
      <transfer_type>B2C</transfer_type>
      <transaction_requirements>{"sample":"requirements"}</transaction_requirements>
    </payer>
    <payer>
      <id>2</id>
      <name>Sample Payer 2</name>
      <currency>EUR</currency>
      <country_iso_code>FRA</country_iso_code>
      <service>MobileWallet</service>
      <transfer_type>B2C</transfer_type>
      <transaction_requirements>{"sample":"requirements"}</transaction_requirements>
    </payer>
  </payers>
</payers_response>
```


**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| **payer** | | |
| id | | |
| name | | |
| currency | | |
| country_iso_code | | |
| service | | |
| transfer_type | | |
| transaction_requirements | | |

# Errors

## Error groups table

| Code | Name | Description |
|---|---|---|
| (100..199) | Systems errors | Transaction could not be processed and was not passed to issuer. |
| (200..299) | Communication errors | Transaction could not be processed properly. Issuer could not be reached or returned invalid data. Errors 230 - 250 need to be reconciled as they might have been processed properly issuer-wise. |
| (300..399) | Input data errors | Transactions cannot be processed due to invalid incoming data in your request. |
| (400..499) | Workflow errors | Workflow errors will occur if you trigger a transaction that is not possible at this time in the workflow, e.g. a refund on a declined transaction. |
| (500..599) | Processing errors | These errors occur when a transaction was declined by the issuer. |
| (600..699) | Risk errors | Risk errors occur when any of the risk management systems will not let the transaction pass through. |
| (700..799) | Tokenization errors | Tokenization related errors. |
| (800..899) | Genesis KYC Services errors | Genesis KYC Services errors |
| (900..999) | Issuer errors | Issuer errors occur when the issuer is unreachable or has other technical problems. If you experience this kind of errors, contact support. |

## Error codes tables

**System Errors**

| Code | Name | Description |
|---|---|---|
| 100 | SystemError | A general system error occurred |
| 101 | MaintenanceError | System is undergoing maintenance, request could not be handled. |

| Code | Name | Description |
|---|---|---|
| 110 | AuthenticationError | Login failed. Check your API credentials. |
| 120 | ConfigurationError | Configuration error occurred, e.g. terminal not config- ured properly. Check terminal settings. |

**Communication Errors**

| Code | Name | Description |
|---|---|---|
| 200 | CommunicationError | Communication with issuer failed, please contact sup- port. |
| 210 | ConnectionError | Connection to issuer could not be established, please contact support. |
| 220 | AccountError | Issuer account data invalid, please contact support. |
| 230 | TimeoutError | Issuer does not respond within given timeframe - please reconcile |
| 240 | ResponseError | Issuer returned invalid response - please reconcile and contact support |
| 250 | ParsingError | Issuer response could not be parsed - please reconcile and contact support. |

**Input Data Errors**

| Code | Name | Description |
|---|---|---|
| 300 | InputDataError | Invalid were data sent to the API. |
| 310 | InvalidTransactionTypeError | Invalid transaction type was passed to API. See transaction types. |
| 320 | InputDataMissingError | Required argument is missing. Check parameters. |
| 330 | InputDataFormatError | Argument passed in invalid format. Check parame- ters. |
| 340 | InputDataInvalidError | Argument passed in valid format but makes no sense (e.g. incorrect country code or currency). Check parameters. |
| 350 | InvalidXmlError | The input XML could not be parsed due to invalid code. Please check XML data. |
| 360 | InvalidContentTypeError | Missing or invalid content type: should be text/xml! |

**Workflow Errors**

| Code | Name | Description |
|---|---|---|
| 400 | WorkflowError | A transaction was triggered that is not possible at this time in the workflow, e.g. a refund on a declined transaction. |
| 410 | ReferenceNotFoundError | Reference transaction was not found. |
| 420 | ReferenceWorkflowError | Wrong Workflow specified. |
| 430 | ReferenceInvalidatedError | Reference transaction already invalidated! |
| 440 | ReferenceMismatchError | Data mismatch with reference, e.g. amount exceeds reference |
| 450 | DoubletTransactionError | Transaction doublet was detected, transaction was blocked. This happens if several transactions with same amount, cardholder, cc number, cvv and expiry date are sent within 5 minutes. |
| 460 | TransactionNotFoundError | The referenced transaction could not be found. |
| 470 | ChargebackNotFoundError | Chargeback not found! |
| 480 | RetrievalRequestNotFoundError | Retrieval Request not found! |
| 490 | FraudReportNotFoundError | Fraud Report not found! |

**Processing Errors**

| Code | Name | Description |
|---|---|---|
| 500 | ProcessingError | Transaction declined by issuer |
| 510 | InvalidCardError | Transaction declined, Credit card number is invalid. |
| 511 | IssuerOctNotEnabledError | OCT not enabled error. |
| 520 | ExpiredCardError | Transaction declined, expiration date not in the future or date invalid. |
| 530 | TransactionPendingError | Transaction pending. |
| 540 | CreditExceededError | Amount excceds credit card limit. |

| Code | Name | Description |
|---|---|---|
| 550 | IssuingError | The voucher could not be issued! |
| 551 | ScaRequiredError | SCA required! |

**Risk Errors**

| Code | Name | Description |
|---|---|---|
| 600 | RiskError | Transaction declined by risk management |
| 601 | InterchangeRejectError | Interchange reject received for transaction! |
| 609 | BinCountryCheckError | Card bin does not match billing country |
| 610 | CardBlacklistError | Card is blacklisted |
| 611 | BinBlacklistError | BIN blacklisted. |
| 612 | CountryBlacklistError | Country blacklisted. |
| 613 | IpBlacklistError | IP address blacklisted. |
| 614 | BlacklistError | value from payment transaction or risk params is blacklisted. |
| 615 | CardWhitelistError | PAN Whitelist Filter blocked the transaction. This filter - like the above one - uses the PAN blacklist (BL) to perform CC number checks against the BL in the DB. This filter however will reject transactions from a CC with a number which is not whitelisted. |
| 620 | CardLimitExceededError | Card limit exceeded configured limits. |
| 621 | TerminalLimitExceededError | Terminal limits exceeded. |
| 622 | ContractLimitExceededError | MID limits exceeded. |
| 623 | CardVelocityExceededError | Velocity by unknown card exceeded! |
| 624 | CardTicketSizeExceededError | Ticketsize by unknown card exceeded! |
| 625 | UserLimitExceededError | User limit exceeded configured limits. |
| 626 | MultipleFailureDetectionError | Found user transaction declined by issuer. Try again later! |
| 627 | CSDetectionError | The CrossSellingFilter blocks duplicated transactions when an approved transaction has been found on another mid. That is, if the transaction has already been processed successfully on a different mid and within the specified time frame, context entity / scope and possibly within the issuer scope (or not), it will be rejected in order to prevent duplicates. |
| 628 | RecurringLimitExceededError | Amount/count by recurring subscription exceeded. |
| 629 | IrisFilterDeclinedError | Transaction declined by risk management. |
| 630 | IrisFilterOnHoldError | Transaction on hold, a manual review will be done |
| 690 | AvsError | Address Verification failed. |
| 691 | MaxMindRiskError | If a transaction is considered high risk by MaxMind minFraud service, a MaxMindRiskError is raised. |
| 692 | ThreatMetrixRiskError | Transaction declined by ThreatMetrix risk module. |
| 693 | IpNotWhitelistedError | Transaction declined by risk management, IP is NOT whitelisted! |
| 694 | DomainBlacklistedError | Transaction declined by risk management, domain is blacklisted! |
| 695 | FraudError | Risk Error: Please contact the risk team! |
| 696 | IbanBlacklistError | Transaction declined by risk management, iban blacklisted! |

**Tokenization Errors**

| Code | Name | Description |
|---|---|---|
| 701 | ConsumerUniquenessError | Consumer with this consumer_id, email combination already exists! |
| 702 | InvalidConsumerError | Consumer not found! |
| 703 | DisabledConsumerError | Consumer is disabled! |
| 700 | TokenizationError | General tokenization error. |
| 710 | TokenizationNotEnabledError | Tokenization is not enabled for the merchant or the terminal! Contact support. |
| 720 | InvalidTokenTypeError | Unsupported token type! |
| 730 | InvalidTokenError | Invalid token! |
| 740 | DetokenizeForbiddenError | Detokenize action is forbidden! |

**KYC Errors**

| Code | Name | Description |
|---|---|---|
| 800 | KycServiceError | General KYC Service Error |
| 801 | DocumentMimeTypeUnsupportedError | Uploaded document MIME type is not supported by KYC provider |
| 802 | InvalidRequestAttributesError | Passed attributes are invalid! |
| 803 | KycServiceNotConfiguredError | KYC Services not configured for Merchant! |
| 804 | KycServiceProviderError | KYC Service provider Error! |
| 805 | KycServiceNotificationError | Notification already received |
| 806 | KycServiceUnacceptableMerchantStateError | Merchant state does not allow using KYC Service API! |

**Remote Errors**

| Code | Name | Description |
|---|---|---|
| 900 | RemoteError | Some error occurred on the issuer. Contact support. |
| 910 | RemoteSystemError | Some error occurred on the issuer |
| 920 | RemoteConfigurationError | Issuer configuration error |
| 930 | RemoteDataError | Some passed data caused an error on the issuer |
| 940 | RemoteWorkflowError | Remote workflow error |
| 950 | RemoteTimeoutError | Issuer has timeouted with clearing network |

> Description can be slightly different for error messages per acquirer, but the error classes will be as documented.

# Client Integrations

There are client libraries and examples for a few programming languages to ease the merchant integration effort:

| Language | Github | Description |
|---|---|---|
| .NET | Genesis .NET | .NET client library |
| Java | Genesis Java | Java client library |
| Java | Genesis Android | Android client library |
| Node.js | Genesis Node | Node.js client library |
| PHP | Genesis PHP | PHP client library |
| Swift | Genesis Swift | iOS client library |

Should you have any questions or suggestions regarding the client libraries and improvements, contact the IT Support team at tech-support@e-comprocessing.com.

You can also fork the repo(s) and send us pull requests directly at our Github account.

# Client-side encryption

The *Client-side Encryption* (CSE) allows merchants to accept payments on their website while encrypting card data in their customer's browser with our JavaScript encryption library.

> The CSE must be used in combination with one of our classic Client Integrations.

To help merchants encrypt all sensitive card data on the customer's side, E-ComProcessing hosts the JavaScript library and merchant's encryption key. In addition, the merchants can decide to host the library by themselves, but we strongly recommend the Client-side encryption library hosted by our services to be used.

If the merchants want to use the Client-Side Encryption (CSE) library, it needs to contact tech-support@e-comprocessing.com in order to enable this feature for them. Then the merchant will be allowed to obtain the public key and the library in the merchant web console.

# Client side

```
<head>
  ...
  <script src="https://{CDN}/encrypto-{VERSION}.js"
          integrity="sha512-lJxHl93A/b8peqxz/mdLj7jD58N2zvHiiYhw8...=="
          crossorigin="anonymous"></script>
</head>

<body>
  <form method="POST" action="/request-payment" id="encrypto-form">
    <div>
      <input type="text" data-encrypted-name="card_number"/>
      <input type="text" data-encrypted-name="card_holder"/>
      ...
      <input type="text" data-encrypted-name="cvv"/>
      ...
      <input type="text" name="country"/>
      <input type="text" name="city"/>
    </div>
    ...
    <input type="submit" value="Pay"/>
  </form>

  <script>
    var publicKey = '...';

    Encrypto.createEncryptedForm(publicKey, {   // Required
      // formId: 'encrypto-form',               // Optional
      // onSubmit: function(form) {             // Optional
      //   console.log(form.fields);
      // }
    })
  </script>
</body>
```

First, the merchant needs to create a payment form integrated with the Client-Side Encryption (CSE) library which can be retrieved from the **Client Side Encryption** panel in the merchant console at the merchant configuration page.

Make sure that payment form contains all required fields for the transaction type which it's going to be used. Consult with our Transactions documentation. Don't forget to replace the form action with the payment handler URL of the merchant's server.

Flag all card input fields for encryption by annotating them with the `data-encrypted-name` attribute. The name attribute should not be used for card input fields. The fields allowed for encryption are `card_holder`, `card_number`, `expiration_month`, `expiration_year`, and `cvv`.

> Use "data-encrypted-name" attribute for card input fields. This technique protects the merchant's server from receiving unencrypted card data and avoids any impact on the transaction security and PCI regulations compliance.

Eventually, the form may have a custom ID attribute. The `formId` option can be used to set any string as an ID for the payment form. Make sure to update the HTML form with the configured option. They both must match.

**Options**

| option | type | description |
|---|---|---|
| formId | string | Use to set custom form ID. Default value `encrypto-form` |
| onSubmit | function | Use to set custom on submit callback |

**PREVENT FORM SUBMIT ACTION**

```
Encrypto.createEncryptedForm(publicKey, {
  onSubmit: function(form) {
    // console.log(form.fields);
  }
})
```

In the case of a single-page application or a form that uses AJAX, maybe it's not desirable the form to reload the page when the payment gets submitted. For that reason, we provide `onSubmit` option. This option gives access to all the form data(including the encrypted fields) and allows to submit it via any AJAX library.

**JAVASCRIPT ONLY**

```
<head>
  ...
  <script src="https://{CDN}/encrypto-{VERSION}.js"
          integrity="sha512-lJxHl93A/b8peqxz/mdLj7jD58N2zvHiiYhw8...=="
          crossorigin="anonymous"></script>
</head>

<body>
  <script>
    var publicKey = '...';
    var cse = Encrypto.createEncryption(publicKey);

    let data = {
      cvv: '123',
      card_number: '42000...'
    };

    var encryptedData = cse.encrypt(data);
  </script>
</body>
```

In case the merchant does not have an HTML form, our library provides HTML-independent encryption. In this scenario, it's important to remember it's the merchant responsibility to make sure the card data is encrypted before sending it to the server.

> Always encrypt the card data before sending to the merchant's server.

The JavaScript-only option can be convenient in case of a more complex single-page application which relays on state management library before sending any data to the server.

---

## Server side

From the merchant's server, an HTTP POST request needs to be made to the gateway API endpoints. The workflow is the same as in the classic Client Integrations. The only difference when the Client-Side Encryption (CSE) library is used is that our gateway will receive the card data encrypted.

> There is no need to change anything in the merchant's server if it's already using any of the Client Integrations.

Do not worry about the decryption. Our gateway will handle the API request as a standard transaction. Make sure to always use the correct public key in the client-side code.

# Shopping Carts

Genesis has a number of shopping cart plugins to ease the merchant integration effort:

| Shopping cart | Github | Marketplace | Description |
|---|---|---|---|
| Magento 1.x CE | Magento CE | | Shopping cart plugin for Magento 1.x CE |
| Magento 2.x CE | Magento 2.x CE | | Shopping cart plugin for Magento 2.x CE |
| OpenCart | OpenCart | OpenCart Extensions | Shopping cart plugin for OpenCart |
| osCommerce | osCommerce | osCommerce Add-Ons | Shopping cart plugin for osCommerce |
| PrestaShop | PrestaShop | | Shopping cart plugin for PrestaShop |
| Shopify | Internal integration | Internal integration | Shopping cart Integration for Shopify |
| WooCommerce | WooCommerce | | Shopping cart plugin for WooCommerce |
| X-Cart | X-Cart | | Shopping cart plugin for X-Cart |
| Zen Cart | Zen Cart | Zen Cart Plugins | Shopping cart plugin for Zen Cart |

Should you have any questions or suggestions regarding the shopping cart plugins and improvements, contact the IT Support team at tech-support@e-comprocessing.com.

You can also fork the repo(s) and send us pull requests directly at our Github account.

# Testing

For testing first login to the gateway admin and create a terminal.

The url for test admin is:

`https://staging.merchant.e-comprocessing.net/`

The api base url for test processing is:

`https://staging.gate.e-comprocessing.net/process/TERMINAL_TOKEN`

The api base url for test single transaction reconciling is:

`https://staging.gate.e-comprocessing.net/reconcile/TERMINAL_TOKEN`

The api base url for test date range reconciling is:

`https://staging.gate.e-comprocessing.net/reconcile/by_date/TERMINAL_TOKEN`

For testing the gateway the following credit card numbers can be used:

| card number | card brand | transaction result |
|---|---|---|
| 4200000000000000 | Visa | successful transaction |
| 4111111111111111 | Visa | transaction declined |
| 5555555555554444 | Master Card | successful transaction |
| 5105105105105100 | Master Card | transaction declined |

For 3DSecure testing the following credit card numbers can be used:

---

## 3DS v1

> For more specifics and flows regarding the 3DSv1 authentication protocol, go to3DSv1 section.

Test Cards The following test cards can be used to return specific responses in each state

| Card Number | Card Brand | Transaction Result |
|---|---|---|
| 4711100000000000 | Visa | 3DSecure enrolled |
| 4012001037461114 | Visa | 3DSecure enrolled failing authentication |
| 4012001036853337 | Visa | 3DSecure unavailable - Card Not Participating |
| 4012001037484447 | Visa | Error in 3DSecure Network in first step of 3DSecure authentication process |
| 4012001036273338 | Visa | Error in 3DSecure Network in second (asynchronous) step of 3DSecure authentication process |
| 5555555555559997 | MasterCard | 3DSecure enrolled |
| 5185540810000019 | MasterCard | 3DSecure enrolled failing authentication |
| 5454545454545454 | MasterCard | 3DSecure unavailable - Card Not Participating |
| 5200828282828210 | MasterCard | Error in 3DSecure Network in first step of 3DSecure authentication process |
| 5204230080000017 | MasterCard | Error in 3DSecure Network in second (asynchronous) step of 3DSecure authentication process |

When redirected to the dummy authentication page you may enter any password you like.

---

## 3DS v2

> For more specifics and flows regarding the 3DSv2 authentication protocol, go to3DSv2 section.

| Scenario | 3DSecure Method | 3DSecure Challenge | Fallback to 3DSv1 | Result | Card Number |
|---|---|---|---|---|---|
| Frictionless | - | - | - | Authenticated | 4012000000060085 |
| Frictionless | Y | - | - | Authenticated | 4066330000000004 |
| Frictionless | - | - | - | Not authenticated | 4111110000000922 |
| Frictionless | Y | - | - | Not authenticated | 4111112232423922 |
| Challenge | - | Y | - | Choose Challenge result | 4918190000000002 |
| Challenge | Y | Y | - | Choose Challenge result | 4938730000000001 |

| Scenario | 3DSecure Method | 3DSecure Challenge | Fallback to 3DSv1 | Result | Card Number |
|---|---|---|---|---|---|
| Fallback to 3DSv1 | - | - | Y | Choose PaRes response | 4901170000000003 |
| Fallback to 3DSv1 | Y | - | Y | Choose PaRes response | 4901164281364345 |

In test mode, successful transaction XML responses will include the following error message: ”TESTMODE: No real money will be transferred!”

# Status Page

statuspage.io is a popular service allowing to track server status, infrastructure notifications, and others.

Note that you can sign up forE-ComProcessing’s status page.

It allows to sign up via email or SMS or both, and receive notifications for our payment services and any planned maintenance windows, upgrades, or similar in the future should the need arise.

# Infrastructure and Uptime

Genesis is hosted in two data centers respectively in Berlin and Amsterdam, and features a state-of-the-art, active-active infrastructure setup. As such, it employs load balancing and failover on the DNS layer, and you should be using and requesting the API nodes and web apps only via their dedicated DNS names.

No hard-coding of IP addresses should be performed on the customers’ systems, as this will prevent the customer to take advantage of the failover in case one of the data centers has issues or throughout maintenance windows and current processing happens through one data center only, however rare this might be. In addition, load balancing of the customers’ volume is also impacted if IP addresses are hardcoded.

Finally, note that the DNS load balancing and failover layer has a TTL of 30 seconds, and will sense any issues returning the right IP addresses to use, for both API nodes and web apps alike, at all times.

As a highly available payment gateway platform, Genesis strives to achieve an uptime SLA of 99.99 percent on a yearly basis.

# Penetration Testing Warning

It is important that merchants read and understand the activities that are explicitly prohibited when using the payment gateway services.

While merchants are encouraged to perform best practice security testing on their own websites and applications, merchants must ensure under all circumstances that scans exclude the payment gateway Web Payment Form (WPF), Processing API, and merchant console.

| Action | prohibited? |
|---|---|
| Penetration testing of any gateway services | Prohibited |
| Load testing | Prohibited. During integration testing, ensure minimum number of requests |
| Exploiting common security vulnerabilities | Prohibited |
| Injecting malicious data | Prohibited |
| Bypassing validation and security checks | Prohibited |
| Subverting ACLs and user permissions | Prohibited |
| Port scanning and service discovery | Prohibited |
| Usage of ping/traceroute | Acceptable for short term debugging purposes |

Note that merchants that do not abide by the above policy will be immediately blacklisted, resulting in terminating access to the WPF, Processing API, and merchant console.

# AVS Status Codes

When sending a transaction for address verification or card verification, one of the following responses from the issuing bank will be received.

| Code | Summary | VISA | MasterCard |
|---|---|---|---|
| A | Partial Match | Address matches, ZIP does not. | Address matches, ZIP does not. |
| B | Partial Match (International Transaction) | Street address match. Postal code not verified because of incompatible formats. (Acquirer sent both street address and postal code) | Street addresses match. Postal code not verified due to incompatible formats. (Acquirer sent both street address and postal code.) |
| C | No Match (International Transaction) | Street address and postal code not verified because of incompatible formats. (Acquirer sent both street address and postal code.) | Street address and postal code not verified due to incompatible formats. (Acquirer sent both street address and postal code.) |
| D | Full Match (International Transaction) | Street addresses and postal code match. | Street addresses and postal codes match. |
| F | Full Match (UK only) | Street address and postal code match. Applies to U.K. only. | Street address and postal code match. Applies to U.K. only. |
| G | Not Supported (International Transaction) | Address information not verified for international transaction. | Address information not verified for international transaction. |
| I | No Match (International Transaction) | Address information not verified. | Address information not verified. |
| M | Full Match (International Transaction) | Street addresses and postal code match. | Street address and postal code match. |
| N | No Match | Neither address nor postal code matches. | Neither address nor postal code matches. |
| P | Partial Match (International Transaction) | Postal codes match. Street address not verified because of incompatible formats. (Acquirer sent both street address and postal code.) | Postal code match. Street address not verified because of incompatible formats. (Acquirer sent both street address and postal code.) |
| R | System Unavailable | Retry, system unable to process. | Retry, system unable to process. |
| S | Not Supported | AVS currently not supported. | AVS currently not supported. |
| U | System Unavailable | No data from issuer/Authorization System. Information not available. | No data from issuer/Authorization System. Information not available. |
| W | Partial Match (US only) | Not applicable. If present, replaced with "Z" by V.I.R Available for U.S. issuers only. | For U.S. addresses, nine-digit postal code matches, address does not; for address outside the U.S., postal code matches, address does not. |
| X | Full Match | Not applicable. | For U.S. addresses, nine-digit postal code and address matches; for addresses outside the U.S., postal code and address match. |
| Y | Full Match | For U.S. addresses, five-digit or nine-digit postal code and address matches. | For U.S. addresses, five-digit postal code and address matches. |
| Z | Partial Match | For U.S. addresses, five-digit or nine-digit postal code matches, address does not. | For U.S. addresses, five-digit postal code matches, address does not. |

AVS status codes are valid for cards issued from United States and United Kingdom. For other countries status message will be returned with error description.

# Level 3 Travel Data

Level 3 travel data is supplied as optional data to the standard API request. If the supplied is valid travel data then the transaction will be processed as a travel transaction and will qualify for the travel Incentive rates. Otherwise the transaction will be processed normally as a regular transaction. Note that the travel data will be stored as part of the transaction in all cases.

Travel data is supported for Authorize, Authorize3D, Capture, Sale, Sale3D, InitRecurringSale, InitRecurringSale3D, RecurringSale.

The following travel types are supported Airline Itinerary Data (AID), Car Rental, Hotel Rental, Ancillary Charges, Misc Charges.

Check the travel related transaction special cases.

## Travel Types

**AIRLINE ITINERARY DATA (AID)**

**Airline Ticket transaction With Airline Itinerary Data (AID)**

**MasterCard**

Contract Merchant Category Code must be 3000-3350, 4511, 4722, 4723 or 5962

Master Card

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Sale');
    $request = $genesis->request();

    $request
        ->set...('')
        ->setTravel('{"ticket"=>{"ticket_number"=>123456789012345, "passenger_name"=>"Test Example", "customer_code"=>1, "issuing_carrier"=>"AAAA", "total_fare"=>5000, "agency_name"=>"Agency", "agency_code"=>"

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.SaleRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    SaleRequest request = new SaleRequest();

    request.set...("");
    request.setTravel("{"ticket"=>{"ticket_number"=>123456789012345, "passenger_name"=>"Test Example", "customer_code"=>1, "issuing_carrier"=>"AAAA", "total_fare"=>5000, "agency_name"=>"Agency", "agency_code"=

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.sale(
  {
    "...": "",
    "travel": {
      "ticket": {
        "ticket_number": 123456789012345,
        "passenger_name": "Test Example",
        "customer_code": 1,
        "issuing_carrier": "AAAA",
        "total_fare": 5000,
        "agency_name": "Agency",
        "agency_code": "AG001"
      },
      "legs": [
        {
          "departure_date": "2018-02-05",
          "carrier_code": 12,
          "service_class": 1,
          "origin_city": "VAR",
          "destination_city": "FRA",
          "stopover_code": 0,
          "fare_basis_code": 1,
          "flight_number": "W6666",
          "departure_time": "11:37",
          "departure_time_segment": "A"
        }
      ]
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <.../>
    <travel>
      <ticket>
        <ticket_number>123456789012345</ticket_number>
        <passenger_name>Test Example</passenger_name>
        <customer_code>1</customer_code>
        <issuing_carrier>AAAA</issuing_carrier>
        <total_fare>5000</total_fare>
        <agency_name>Agency</agency_name>
        <agency_code>AG001</agency_code>
      </ticket>
      <legs>
        <leg>
          <departure_date>2018-02-05</departure_date>
          <carrier_code>12</carrier_code>
          <service_class>1</service_class>
          <origin_city>VAR</origin_city>
          <destination_city>FRA</destination_city>
          <stopover_code>0</stopover_code>
          <fare_basis_code>1</fare_basis_code>
          <flight_number>W6666</flight_number>
          <departure_time>11:37</departure_time>
          <departure_time_segment>A</departure_time_segment>
        </leg>
      </legs>
    </travel>
  </payment_transaction>'
```

Master Card Multiple Legs

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Sale');
    $request = $genesis->request();

    $request
        ->set...('')
        ->setTravel('{"ticket"=>{"ticket_number"=>123456789012345, "passenger_name"=>"Test Example", "customer_code"=>1, "issuing_carrier"=>"AAAA", "total_fare"=>5000, "agency_name"=>"Agency", "agency_code"=>"

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.SaleRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    SaleRequest request = new SaleRequest();

    request.set...("");
    request.setTravel("{"ticket"=>{"ticket_number"=>123456789012345, "passenger_name"=>"Test Example", "customer_code"=>1, "issuing_carrier"=>"AAAA", "total_fare"=>5000, "agency_name"=>"Agency", "agency_code"=

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.sale(
  {
    "...": "",
    "travel": {
      "ticket": {
        "ticket_number": 123456789012345,
        "passenger_name": "Test Example",
        "customer_code": 1,
        "issuing_carrier": "AAAA",
        "total_fare": 5000,
        "agency_name": "Agency",
        "agency_code": "AG001"
      },
      "legs": [
        {
          "departure_date": "2018-02-05",
          "carrier_code": 12,
          "service_class": 1,
          "origin_city": "VAR",
          "destination_city": "FRA",
          "stopover_code": 0,
          "fare_basis_code": 1,
          "flight_number": "W6666",
          "departure_time": "11:37",
          "departure_time_segment": "A"
        },
        {
          "departure_date": "2018-02-05",
          "carrier_code": 12,
          "service_class": 1,
          "origin_city": "VAR",
          "destination_city": "BER",
          "stopover_code": 0,
          "fare_basis_code": 1,
          "flight_number": "W6666",
          "departure_time": "11:37",
          "departure_time_segment": "A"
        }
      ]
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <.../>
   <travel>
     <ticket>
       <ticket_number>123456789012345</ticket_number>
       <passenger_name>Test Example</passenger_name>
       <customer_code>1</customer_code>
       <issuing_carrier>AAAA</issuing_carrier>
       <total_fare>5000</total_fare>
       <agency_name>Agency</agency_name>
       <agency_code>AG001</agency_code>
     </ticket>
     <legs>
       <leg>
         <departure_date>2018-02-05</departure_date>
         <carrier_code>12</carrier_code>
         <service_class>1</service_class>
         <origin_city>VAR</origin_city>
         <destination_city>FRA</destination_city>
         <stopover_code>0</stopover_code>
         <fare_basis_code>1</fare_basis_code>
         <flight_number>W6666</flight_number>
         <departure_time>11:37</departure_time>
         <departure_time_segment>A</departure_time_segment>
       </leg>
       <leg>
         <departure_date>2018-02-05</departure_date>
         <carrier_code>12</carrier_code>
         <service_class>1</service_class>
         <origin_city>VAR</origin_city>
         <destination_city>BER</destination_city>
         <stopover_code>0</stopover_code>
         <fare_basis_code>1</fare_basis_code>
         <flight_number>W6666</flight_number>
         <departure_time>11:37</departure_time>
         <departure_time_segment>A</departure_time_segment>
       </leg>
     </legs>
   </travel>
 </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| **travel** | required | | |
| **ticket** | required* | | |
| ticket_number | required* | String(15) | The number on the ticket. |
| passenger_name | required* | String(29) | The name of the passenger. May be the cardholder name if the passenger name is unavailable. Must not be blank. |
| customer_code | required* | String(17) | The customer code. Internal Reference. |
| issuing_carrier | optional | String(4) | Contains the standard abbreviation for the airline or railway carrier issuing the ticket. |
| total_fare | required* | Integer | Total amount of the ticket and should equal the amount of the transaction. |
| agency_name | optional | String(30) | An entry should be supplied if a travel agency issued the ticket. |
| agency_code | optional | String(8) | An entry should be supplied if a travel agency issued the ticket. |
| confirmation_information | required* | String(474) | Confirmation Information |
| date_of_issue | required* | String(10) | Date Of Issue |
| **legs** | required* | | Max legs 10 |
| **leg** | required* | | |
| departure_date | required | String(10) | The departure date. Date can be in future. |
| arrival_date | required* | String(10) | The arrival date. Date can be in future. |
| carrier_code | required* | String(2) | Contains the standard abbreviation for the airline or railway carrier issuing the ticket. This should not contain all spaces or zeros. Code indicating name of carrier. |
| service_class | required* | String(1) | The service type. i.e. Coach, First Class. Required for reduced interchange. |
| origin_city | required* | String(3) | The originating airport name‘s standard abbreviation. This should not contain all spaces or zeroes. |
| destination_city | required* | String(3) | The destination airport or railway name‘s standard abbreviation. |
| stopover_code | required* | String(1) | A code indicating whether there was a direct or a non-direct flight or route on the same ticket number. Allowed values: 0, 1 |

| Parameter | Required | Format | Description |
|---|---|---|---|
| fare_basis_code | optional | String(6) | A code that carriers assign to a particular ticket type, such as business class or discounted/ non-re fundable. |
| flight_number | optional | String(5) | The number that the operating or marketing carrier assigned. |
| departure_time | optional | String(5) | The time of departure provided by the airline or railway, per trip leg. |
| departure_time_segment | optional | String(1) | Departure Time Segment. Allowed values: A, P |
| **taxes** | optional | | Max taxes 10 |
| **tax** | optional | | |
| fee_amount | required* | Integer | Fee Amount |
| fee_type | required* | String(8) | Fee Type |

`required* = conditionally required`

**Visa**

Contract Merchant Category Code must be 3000-3350, 4511, 4722, 4723 or 5962

Visa

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Sale');
    $request = $genesis->request();

    $request
        ->set...('')
        ->setTravel('{"ticket"=>{"ticket_number"=>12345, "passenger_name"=>"Emil Example", "customer_code"=>1, "restricted_ticket_indicator"=>1, "agency_name"=>"Agency", "agency_code"=>"AG001", "confirmation_i

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.SaleRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    SaleRequest request = new SaleRequest();

    request.set...("");
    request.setTravel("{"ticket"=>{"ticket_number"=>12345, "passenger_name"=>"Emil Example", "customer_code"=>1, "restricted_ticket_indicator"=>1, "agency_name"=>"Agency", "agency_code"=>"AG001", "confirmatio

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.sale(
  {
    "...": "",
    "travel": {
      "ticket": {
        "ticket_number": 12345,
        "passenger_name": "Emil Example",
        "customer_code": 1,
        "restricted_ticket_indicator": 1,
        "agency_name": "Agency",
        "agency_code": "AG001",
        "confirmation_information": "Confirmation",
        "date_of_issue": "2018-02-01"
      },
      "legs": [
        {
          "departure_date": "2018-02-01",
          "carrier_code": 2,
          "service_class": 3,
          "origin_city": "SOF",
          "destination_city": "VAR",
          "stopover_code": 0,
          "fare_basis_code": 1,
          "flight_number": "W6666"
        }
      ],
      "taxes": [
        {
          "fee_amount": 1000,
          "fee_type": "Airport tax"
        }
      ]
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <.../>
    <travel>
      <ticket>
        <ticket_number>12345</ticket_number>
        <passenger_name>Emil Example</passenger_name>
        <customer_code>1</customer_code>
        <restricted_ticket_indicator>1</restricted_ticket_indicator>
        <agency_name>Agency</agency_name>
        <agency_code>AG001</agency_code>
        <confirmation_information>Confirmation</confirmation_information>
        <date_of_issue>2018-02-01</date_of_issue>
      </ticket>
      <legs>
        <leg>
          <departure_date>2018-02-01</departure_date>
          <carrier_code>2</carrier_code>
          <service_class>3</service_class>
          <origin_city>SOF</origin_city>
          <destination_city>VAR</destination_city>
          <stopover_code>0</stopover_code>
          <fare_basis_code>1</fare_basis_code>
          <flight_number>W6666</flight_number>
        </leg>
      </legs>
      <taxes>
        <taxe>
          <fee_amount>1000</fee_amount>
          <fee_type>Airport tax</fee_type>
        </taxe>
      </taxes>
    </travel>
  </payment_transaction>'
```

Visa Multiple Legs

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Sale');
    $request = $genesis->request();

    $request
        ->set...('')
        ->setTravel('{"ticket"=>{"ticket_number"=>123456789012345, "passenger_name"=>"Test Example", "customer_code"=>1, "restricted_ticket_indicator"=>1, "agency_name"=>"Agency", "agency_code"=>"AG001"}, "leg

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.SaleRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    SaleRequest request = new SaleRequest();

    request.set...("");
    request.setTravel("{"ticket"=>{"ticket_number"=>123456789012345, "passenger_name"=>"Test Example", "customer_code"=>1, "restricted_ticket_indicator"=>1, "agency_name"=>"Agency", "agency_code"=>"AG001"}, "

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.sale(
  {
    "...": "",
    "travel": {
      "ticket": {
        "ticket_number": 123456789012345,
        "passenger_name": "Test Example",
        "customer_code": 1,
        "restricted_ticket_indicator": 1,
        "agency_name": "Agency",
        "agency_code": "AG001"
      },
      "legs": [
        {
          "departure_date": "2018-02-01",
          "carrier_code": 2,
          "service_class": 3,
          "origin_city": "SOF",
          "destination_city": "VAR",
          "stopover_code": 0,
          "fare_basis_code": 1,
          "flight_number": "W6666"
        },
        {
          "departure_date": "2018-02-01",
          "carrier_code": 2,
          "service_class": 3,
          "origin_city": "VAR",
          "destination_city": "FRA",
          "stopover_code": 0,
          "fare_basis_code": 1,
          "flight_number": "W6366"
        }
      ]
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <.../>
    <travel>
      <ticket>
        <ticket_number>123456789012345</ticket_number>
        <passenger_name>Test Example</passenger_name>
        <customer_code>1</customer_code>
        <restricted_ticket_indicator>1</restricted_ticket_indicator>
        <agency_name>Agency</agency_name>
        <agency_code>AG001</agency_code>
      </ticket>
      <legs>
        <leg>
          <departure_date>2018-02-01</departure_date>
          <carrier_code>2</carrier_code>
          <service_class>3</service_class>
          <origin_city>SOF</origin_city>
          <destination_city>VAR</destination_city>
          <stopover_code>0</stopover_code>
          <fare_basis_code>1</fare_basis_code>
          <flight_number>W6666</flight_number>
        </leg>
        <leg>
          <departure_date>2018-02-01</departure_date>
          <carrier_code>2</carrier_code>
          <service_class>3</service_class>
          <origin_city>VAR</origin_city>
          <destination_city>FRA</destination_city>
          <stopover_code>0</stopover_code>
          <fare_basis_code>1</fare_basis_code>
          <flight_number>W6366</flight_number>
        </leg>
      </legs>
    </travel>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| **travel** | required | | |
| **ticket** | required* | | |
| ticket_number | required* | String(15) | The number on the ticket. |
| passenger_name | required* | String(29) | The name of the passenger. May be the cardholder name if the passenger name is unavailable. Must not be blank. |
| customer_code | required* | String(17) | The customer code. Internal Reference. |
| confirmation_information | required* | String(474) | Confirmation Information |
| date_of_issue | required* | String(10) | Date Of Issue |
| restricted_ticket_indicator | optional | String(1) | Space or 0 = No restriction, 1 = Restriction; Allowed values: Empty String, 0, 1 |
| agency_name | optional | String(30) | An entry should be supplied if a travel agency issued the ticket. |
| agency_code | optional | String(8) | An entry should be supplied if a travel agency issued the ticket. |
| **legs** | required* | | Max legs 10 |
| **leg** | required* | | |
| departure_date | required | String(10) | The departure date. Date can be in future. |
| arrival_date | required* | String(10) | The arrival date. Date can be in future. |
| origin_city | required* | String(3) | The originating airport name‘s standard abbreviation. This should not contain all spaces or zeroes. |
| carrier_code | required* | String(2) | Contains the standard abbreviation for the airline or railway carrier issuing the ticket. This should not contain all spaces or zeros. Code indicating name of carrier. |
| service_class | required* | String(1) | The service type. i.e. Coach, First Class. Required for reduced interchange. |
| stopover_code | required* | String(1) | A code indicating whether there was a direct or a non-direct flight or route on the same ticket number. Allowed values: 0, 1 |
| destination_city | required* | String(3) | The destination airport or railway name‘s standard abbreviation. |
| fare_basis_code | optional | String(6) | A code that carriers assign to a particular ticket type, such as business class or discounted/ non-re fundable. |
| flight_number | optional | String(5) | The number that the operating or marketing carrier assigned. |
| **taxes** | optional | | Max taxes 10 |
| **tax** | optional | | |
| fee_amount | required* | Integer | Fee Amount |
| fee_type | required* | String(8) | Fee Type |

`required* = conditionally required`

**CAR RENTAL**

**MasterCard**

Contract Merchant Category Code must be 3351-3500, 4722, 4723, 5962, 7512, 7513, 7519

Master Card

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Sale');
    $request = $genesis->request();

    $request
        ->set...('')
        ->setTravel('{"rentals"=>{"car_rental"=>{"purchase_identifier"=>12478, "class_id"=>3, "pickup_date"=>"2018-02-05", "renter_name"=>"Emil Example", "return_city"=>"Varna", "return_state"=>"VAR", "return_

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.SaleRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    SaleRequest request = new SaleRequest();

    request.set...("");
    request.setTravel("{"rentals"=>{"car_rental"=>{"purchase_identifier"=>12478, "class_id"=>3, "pickup_date"=>"2018-02-05", "renter_name"=>"Emil Example", "return_city"=>"Varna", "return_state"=>"VAR", "retu

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.sale(
  {
    "...": "",
    "travel": {
      "rentals": {
        "car_rental": {
          "purchase_identifier": 12478,
          "class_id": 3,
          "pickup_date": "2018-02-05",
          "renter_name": "Emil Example",
          "return_city": "Varna",
          "return_state": "VAR",
          "return_country": "BGR",
          "return_date": "2018-02-06",
          "renter_return_location_id": 12478,
          "customer_code": 1
        }
      }
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <.../>
   <travel>
     <rentals>
       <car_rental>
         <purchase_identifier>12478</purchase_identifier>
         <class_id>3</class_id>
         <pickup_date>2018-02-05</pickup_date>
         <renter_name>Emil Example</renter_name>
         <return_city>Varna</return_city>
         <return_state>VAR</return_state>
         <return_country>BGR</return_country>
         <return_date>2018-02-06</return_date>
         <renter_return_location_id>12478</renter_return_location_id>
         <customer_code>1</customer_code>
       </car_rental>
     </rentals>
   </travel>
 </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| **travel** | required | | |
| **rentals** | required | | |
| **car_rental** | required | | |
| purchase_identifier | required* | String(9) | Rental Agreement Number / Hotel Folio Number. |
| class_id | required* | String(4) | The car rental classification. Allowed values: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 9999 |
| pickup_date | required | String(10) | Car rental Pick-up date. |
| renter_name | required* | String(20) | The Renter Name |
| return_city | required* | String(18) | The Rental Return City |
| return_state | required* | String(3) | The Rental Return State |
| return_country | required* | String(3) | The Rental Return Country |
| return_date | required | String(10) | Car Rental return date |
| renter_return_location_id | required* | String(10) | Expenses or Car Rental code, Address, phone number, etc. Identifying Rental Return Location. |
| customer_code | required* | String(17) | The customer code. Internal Reference. |

`required* = conditionally required`

**Visa**

Contract Merchant Category Code must be 3351-3500, 4722, 4723, 5962, 7512, 7513, 7519

Visa

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Sale');
    $request = $genesis->request();

    $request
        ->set...('')
        ->setTravel('{"rentals"=>{"car_rental"=>{"purchase_identifier"=>12478, "class_id"=>3, "pickup_date"=>"2018-02-05", "renter_name"=>"Emil Example", "return_city"=>"Varna", "return_state"=>"VAR", "return_

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.SaleRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    SaleRequest request = new SaleRequest();

    request.set...("");
    request.setTravel("{"rentals"=>{"car_rental"=>{"purchase_identifier"=>12478, "class_id"=>3, "pickup_date"=>"2018-02-05", "renter_name"=>"Emil Example", "return_city"=>"Varna", "return_state"=>"VAR", "retu

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.sale(
  {
    "...": "",
    "travel": {
      "rentals": {
        "car_rental": {
          "purchase_identifier": 12478,
          "class_id": 3,
          "pickup_date": "2018-02-05",
          "renter_name": "Emil Example",
          "return_city": "Varna",
          "return_state": "VAR",
          "return_country": "BGR",
          "return_date": "2018-02-06",
          "renter_return_location_id": 12478,
          "customer_code": 1
        }
      }
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <.../>
    <travel>
      <rentals>
        <car_rental>
          <purchase_identifier>12478</purchase_identifier>
          <class_id>3</class_id>
          <pickup_date>2018-02-05</pickup_date>
          <renter_name>Emil Example</renter_name>
          <return_city>Varna</return_city>
          <return_state>VAR</return_state>
          <return_country>BGR</return_country>
          <return_date>2018-02-06</return_date>
          <renter_return_location_id>12478</renter_return_location_id>
          <customer_code>1</customer_code>
        </car_rental>
      </rentals>
    </travel>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| **travel** | required | | |
| **rentals** | required | | |
| **car_rental** | required | | |
| purchase_identifier | optional | String(25) | Rental Agreement Number / Hotel Folio Number |
| pickup_date | required* | String(10) | Car rental Pick-up date. |
| return_date | required* | String(10) | Car rental Return date. |
| extra_charges | optional | Array(6) | Additional charges added to customer bill after check-out. Each position can be used to indicate a type of charge; Allowed values: 1, 2, 3, 4, 5 |
| no_show_indicator | optional | String(1) | No show indicator; Allowed values: 0, 1 |

`required* = conditionally required`

**HOTEL RENTAL**

**MasterCard**

Contract Merchant Category Code must be 3501-3999, 4722, 4723, 5962, 7011

Master Card

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Sale');
    $request = $genesis->request();

    $request
        ->set...('')
        ->setTravel('{"rentals"=>{"hotel_rental"=>{"purchase_identifier"=>12478, "arrival_date"=>3, "departure_date"=>"2018-02-05", "customer_code"=>1}}}');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.SaleRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    SaleRequest request = new SaleRequest();

    request.set...("");
    request.setTravel("{"rentals"=>{"hotel_rental"=>{"purchase_identifier"=>12478, "arrival_date"=>3, "departure_date"=>"2018-02-05", "customer_code"=>1}}}");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.sale(
  {
    "...": "",
    "travel": {
      "rentals": {
        "hotel_rental": {
          "purchase_identifier": 12478,
          "arrival_date": 3,
          "departure_date": "2018-02-05",
          "customer_code": 1
        }
      }
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <.../>
    <travel>
      <rentals>
        <hotel_rental>
          <purchase_identifier>12478</purchase_identifier>
          <arrival_date>3</arrival_date>
          <departure_date>2018-02-05</departure_date>
          <customer_code>1</customer_code>
        </hotel_rental>
      </rentals>
    </travel>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| **travel** | required | | |
| **rentals** | required | | |
| **hotel_rental** | required | | |
| purchase_identifier | required* | String(10) | Rental Agreement Number / Hotel Folio Number. |
| arrival_date | required | String(10) | Hotel check-in date. |
| departure_date | required | String(10) | The departure date. Date can be in future. |
| customer_code | required* | String(17) | The customer code. Internal Reference. |

`required* = conditionally required`

**Visa**

Contract Merchant Category Code must be 3501-3999, 4722, 4723, 5962, 7011

Visa

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Sale');
    $request = $genesis->request();

    $request
        ->set...('')
        ->setTravel('{"rentals"=>{"hotel_rental"=>{"purchase_identifier"=>2, "arrival_date"=>"2018-02-01", "extra_charges"=>467, "no_show_indicator"=>1}}}');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.SaleRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    SaleRequest request = new SaleRequest();

    request.set...("");
    request.setTravel("{"rentals"=>{"hotel_rental"=>{"purchase_identifier"=>2, "arrival_date"=>"2018-02-01", "extra_charges"=>467, "no_show_indicator"=>1}}}");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.sale(
  {
    "...": "",
    "travel": {
      "rentals": {
        "hotel_rental": {
          "purchase_identifier": 2,
          "arrival_date": "2018-02-01",
          "extra_charges": 467,
          "no_show_indicator": 1
        }
      }
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```bash
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <.../>
    <travel>
      <rentals>
        <hotel_rental>
          <purchase_identifier>2</purchase_identifier>
          <arrival_date>2018-02-01</arrival_date>
          <extra_charges>467</extra_charges>
          <no_show_indicator>1</no_show_indicator>
        </hotel_rental>
      </rentals>
    </travel>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| **travel** | required | | |
| **rentals** | required | | |
| **hotel_rental** | required | | |

| Parameter | Required | Format | Description |
|---|---|---|---|
| purchase_identifier | optional | String(25) | Rental Agreement Number / Hotel Folio Number |
| arrival_date | required* | String(10) | Hotel rental Pick-up date. |
| departure_date | required* | String(10) | Hotel rental Departure date. |
| extra_charges | optional | Array(6) | Additional charges added to customer bill after check-out. Each position can be used to indicate a type of charge. Allowed values: 2, 3, 4, 5, 6, 7 |
| no_show_indicator | optional | String(1) | No show indicator; Allowed values: 0, 1 |

`required* = conditionally required`

#### ANCILLARY CHARGES

### Ancillary Charges

Charges/fees related to the ticket. These transactions are processed on a separate transaction, referenced to Airline transaction with AID.

### MasterCard

Used to identify only Baggage Charges.

Contract Merchant Category Code must be 3000-3350, 4511, 4722, 4723 or 5962

Master Card

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Sale');
    $request = $genesis->request();

    $request
        ->set...('')
        ->setTravel('{"ticket"=>{"ticket_reference_id"=>"8b7e3575e5605ea7e1895707a3e92837"}, "charges"=>{"charge"=>{"type"=>"BG"}}}');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.SaleRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    SaleRequest request = new SaleRequest();

    request.set...("");
    request.setTravel("{"ticket"=>{"ticket_reference_id"=>"8b7e3575e5605ea7e1895707a3e92837"}, "charges"=>{"charge"=>{"type"=>"BG"}}}");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.sale(
  {
    "...": "",
    "travel": {
      "ticket": {
        "ticket_reference_id": "8b7e3575e5605ea7e1895707a3e92837"
      },
      "charges": {
        "charge": {
          "type": "BG"
        }
      }
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <.../>
    <travel>
      <ticket>
        <ticket_reference_id>8b7e3575e5605ea7e1895707a3e92837</ticket_reference_id>
      </ticket>
      <charges>
        <charge>
          <type>BG</type>
        </charge>
      </charges>
    </travel>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| **travel** | required | | |
| **ticket** | required | | |
| ticket_reference_id | required | String(32) | Unique id of the ticket transaction |
| **charges** | required | | |
| **charge** | required | | |
| type | required | String(2) | This field will contain the Service Category Code for the primary type of service that has been provided. Allowed values: BG |

`required* = conditionally required`

**Visa**

Used to identify charges for a number of ancillary services such as ticket upgrades, baggage fee, food & beverage purchases which are not purchased as part of the original ticket. Also used for charges/fees related to partial airline ticket refunds or ticket cancellations.

Contract Merchant Category Code must be 3000-3350, 4511, 4722, 4723 or 5962

Visa

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Sale');
    $request = $genesis->request();

    $request
        ->set...('')
        ->setTravel('{"ticket"=>{"ticket_reference_id"=>"8b7e3575e5605ea7e1895707a3e92837", "ticket_document_number"=>1111, "issued_with_ticket_number"=>12321}, "charges"=>{"charge"=>{"type"=>"BF", "sub_type"=

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.SaleRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    SaleRequest request = new SaleRequest();

    request.set...("");
    request.setTravel("{"ticket"=>{"ticket_reference_id"=>"8b7e3575e5605ea7e1895707a3e92837", "ticket_document_number"=>1111, "issued_with_ticket_number"=>12321}, "charges"=>{"charge"=>{"type"=>"BF", "sub_typ

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.sale(
  {
    "...": "",
    "travel": {
      "ticket": {
        "ticket_reference_id": "8b7e3575e5605ea7e1895707a3e92837",
        "ticket_document_number": 1111,
        "issued_with_ticket_number": 12321
      },
      "charges": {
        "charge": {
          "type": "BF",
          "sub_type": "BG"
        }
      }
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <.../>
    <travel>
      <ticket>
        <ticket_reference_id>8b7e3575e5605ea7e1895707a3e92837</ticket_reference_id>
        <ticket_document_number>1111</ticket_document_number>
        <issued_with_ticket_number>12321</issued_with_ticket_number>
      </ticket>
      <charges>
        <charge>
          <type>BF</type>
          <sub_type>BG</sub_type>
        </charge>
      </charges>
    </travel>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| **travel** | required | | |
| **ticket** | required | | |
| ticket_reference_id | required | String(32) | Unique id of the ticket transaction |
| ticket_document_number | required | String(15) | This field will contain the form number assigned by the carrier for the transaction. |
| issued_with_ticket_number | required | String(15) | If this purchase has a connection or relationship to another purchase, such as baggage fee for a passenger transport ticket, this field must contain the document number for the other purchase. |
| **charges** | required | | |
| **charge** | required | | |
| type | required | String(2) | This field will contain the Service Category Code for the primary type of service that has been provided. Allowed values: BF, BG, CF, CG, CO, FF, GF, GT, IE, LG, MD, ML, OT, PA, PT, SA, SB, SF, ST, TS, UN, UP, WI |
| sub_type | required | String(2) | This field will contain the Service Category Code for the secondary type of service that has been provided Allowed values: BF, BG, CF, CG, CO, FF, GF, GT, IE, LG, MD, ML, OT, PA, PT, SA, SB, SF, ST, TS, UN, UP, WI |

`required* = conditionally required`

**MISCELLANEOUS CHARGES**

Miscellaneous charges related to the travel, but not related to the ticket.

**MasterCard**

Contract Merchant Category Code must be 3000-3350, 4511, 4722, 4723 or 5962

Master Card

Request

```
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Sale');
    $request = $genesis->request();

    $request
        ->set...('')
        ->setTravel('{"ticket"=>{"ticket_reference_id"=>"8b7e3575e5605ea7e1895707a3e92837"}, "charges"=>{"charge"=>{"type"=>"MISC"}}}');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.SaleRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    SaleRequest request = new SaleRequest();

    request.set...("");
    request.setTravel("{"ticket"=>{"ticket_reference_id"=>"8b7e3575e5605ea7e1895707a3e92837"}, "charges"=>{"charge"=>{"type"=>"MISC"}}}");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.sale(
  {
    "...": "",
    "travel": {
      "ticket": {
        "ticket_reference_id": "8b7e3575e5605ea7e1895707a3e92837"
      },
      "charges": {
        "charge": {
          "type": "MISC"
        }
      }
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```bash
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <.../>
    <travel>
      <ticket>
        <ticket_reference_id>8b7e3575e5605ea7e1895707a3e92837</ticket_reference_id>
      </ticket>
      <charges>
        <charge>
          <type>MISC</type>
        </charge>
      </charges>
    </travel>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| **travel** | required | | |
| **ticket** | required | | |
| ticket_reference_id | required | String(32) | Unique id of the ticket transaction |

| Parameter | Required | Format | Description |
|---|---|---|---|
| **charges** | required | | |
| **charge** | required | | |
| type | required | String(4) | This field will contain the Service Category Code for the primary type of service that has been provided. Allowed values: MISC |

`required* = conditionally required`

**Visa**

Contract Merchant Category Code must be 3000-3350, 4511, 4722, 4723 or 5962

Visa

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Sale');
    $request = $genesis->request();

    $request
        ->set...('')
        ->setTravel('{"ticket"=>{"ticket_reference_id"=>"8b7e3575e5605ea7e1895707a3e92837"}, "charges"=>{"charge"=>{"type"=>"MISC"}}}');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.SaleRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    SaleRequest request = new SaleRequest();

    request.set...("");
    request.setTravel("{"ticket"=>{"ticket_reference_id"=>"8b7e3575e5605ea7e1895707a3e92837"}, "charges"=>{"charge"=>{"type"=>"MISC"}}}");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.sale(
  {
    "...": "",
    "travel": {
      "ticket": {
        "ticket_reference_id": "8b7e3575e5605ea7e1895707a3e92837"
      },
      "charges": {
        "charge": {
          "type": "MISC"
        }
      }
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <.../>
    <travel>
      <ticket>
        <ticket_reference_id>8b7e3575e5605ea7e1895707a3e92837</ticket_reference_id>
      </ticket>
      <charges>
        <charge>
          <type>MISC</type>
        </charge>
      </charges>
    </travel>
  </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| **travel** | required | | |
| **ticket** | required | | |
| ticket_reference_id | required | String(32) | Unique id of the ticket transaction |
| **charges** | required | | |
| **charge** | required | | |
| type | required | String(4) | This field will contain the Service Category Code for the primary type of service that has been provided. Allowed values: MISC |

`required* = conditionally required`

---

# Special Cases

**TRAVEL AUTHORIZE (3D) AND CAPTURE**

The Capture travel data is always merged with the Authorize travel data and overrides the Authorization fields (where they are present in both transactions) before validating. This makes the required travel data for Authorizations optional. This logic is applied for all Travel Types. Because of this, there are 4 scenarios for submitting travel Authorization and Capture.

**Travel Authorize (3D) and Travel Capture**

In this scenario the Authorize and Capture transaction requests are submitted with valid travel data. The travel data that will be used for the transaction processing is the data submitted with the Capture.

Example Travel Authorize

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Authorize');
    $request = $genesis->request();

    $request
        ->setTransactionId('1915c52b28ff29edb12bc617ac102a03')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('4200000000000000')
        ->setCvv('834')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setTravel('{"ticket"=>{"ticket_number"=>12345678123456, "passenger_name"=>"Passenger 01", "customer_code"=>123, "restricted_ticket_indicator"=>0, "agency_name"=>"Agency", "agency_code"=>"AG001"}, "le

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.AuthorizeRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    AuthorizeRequest request = new AuthorizeRequest();

    request.setTransactionId("1915c52b28ff29edb12bc617ac102a03");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("4200000000000000");
    request.setCvv("834");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setTravel("{"ticket"=>{"ticket_number"=>12345678123456, "passenger_name"=>"Passenger 01", "customer_code"=>123, "restricted_ticket_indicator"=>0, "agency_name"=>"Agency", "agency_code"=>"AG001"}, "

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.authorize(
  {
    "transaction_id": "1915c52b28ff29edb12bc617ac102a03",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "4200000000000000",
    "cvv": "834",
    "expiration_month": "12",
    "expiration_year": 2021,
    "travel": {
      "ticket": {
        "ticket_number": 12345678123456,
        "passenger_name": "Passenger 01",
        "customer_code": 123,
        "restricted_ticket_indicator": 0,
        "agency_name": "Agency",
        "agency_code": "AG001"
      },
      "legs": [
        {
          "departure_date": "2017-03-10",
          "carrier_code": "VX",
          "service_class": "J",
          "origin_city": "DUB",
          "destination_city": "ATL",
          "stopover_code": 1,
          "fare_basis_code": 0,
          "flight_number": ""
        }
      ]
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```curl
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>authorize</transaction_type>
    <transaction_id>1915c52b28ff29edb12bc617ac102a03</transaction_id>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>100</amount>
    <currency>USD</currency>
    <card_holder>Travis Pastrana</card_holder>
    <card_number>4200000000000000</card_number>
    <cvv>834</cvv>
    <expiration_month>12</expiration_month>
    <expiration_year>2021</expiration_year>
    <travel>
      <ticket>
        <ticket_number>12345678123456</ticket_number>
        <passenger_name>Passenger 01</passenger_name>
        <customer_code>123</customer_code>
        <restricted_ticket_indicator>0</restricted_ticket_indicator>
        <agency_name>Agency</agency_name>
        <agency_code>AG001</agency_code>
      </ticket>
      <legs>
        <leg>
          <departure_date>2017-03-10</departure_date>
          <carrier_code>VX</carrier_code>
          <service_class>J</service_class>
          <origin_city>DUB</origin_city>
          <destination_city>ATL</destination_city>
          <stopover_code>1</stopover_code>
          <fare_basis_code>0</fare_basis_code>
          <flight_number/>
        </leg>
      </legs>
    </travel>
  </payment_transaction>'
```

Example Travel Capture

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Capture');
    $request = $genesis->request();

    $request
        ->setTransactionId('7fcc6097153beeec1dd1e71f643198d8')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setReferenceId('1f2ee425e7c3159c60743098071771eb')
        ->setTravel('{"ticket"=>{"ticket_number"=>12345678123456, "passenger_name"=>"Passenger 01", "customer_code"=>"", "restricted_ticket_indicator"=>0, "agency_name"=>"New Agency", "agency_code"=>"AGN001"},

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.CaptureRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    CaptureRequest request = new CaptureRequest();

    request.setTransactionId("7fcc6097153beeec1dd1e71f643198d8");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setReferenceId("1f2ee425e7c3159c60743098071771eb");
    request.setTravel("{"ticket"=>{"ticket_number"=>12345678123456, "passenger_name"=>"Passenger 01", "customer_code"=>"", "restricted_ticket_indicator"=>0, "agency_name"=>"New Agency", "agency_code"=>"AGN001"

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.capture(
  {
    "transaction_id": "7fcc6097153beeec1dd1e71f643198d8",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "reference_id": "1f2ee425e7c3159c60743098071771eb",
    "travel": {
      "ticket": {
        "ticket_number": 12345678123456,
        "passenger_name": "Passenger 01",
        "customer_code": "",
        "restricted_ticket_indicator": 0,
        "agency_name": "New Agency",
        "agency_code": "AGN001"
      },
      "legs": [
        {
          "departure_date": "2017-03-10",
          "carrier_code": "VX",
          "service_class": "J",
          "origin_city": "DUB",
          "destination_city": "ATL",
          "stopover_code": 1,
          "fare_basis_code": 0,
          "flight_number": ""
        }
      ]
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>capture</transaction_type>
    <transaction_id>7fcc6097153beeec1dd1e71f643198d8</transaction_id>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>100</amount>
    <currency>USD</currency>
    <reference_id>1f2ee425e7c3159c60743098071771eb</reference_id>
    <travel>
      <ticket>
        <ticket_number>12345678123456</ticket_number>
        <passenger_name>Passenger 01</passenger_name>
        <customer_code/>
        <restricted_ticket_indicator>0</restricted_ticket_indicator>
        <agency_name>New Agency</agency_name>
        <agency_code>AGN001</agency_code>
      </ticket>
      <legs>
        <leg>
          <departure_date>2017-03-10</departure_date>
          <carrier_code>VX</carrier_code>
          <service_class>J</service_class>
          <origin_city>DUB</origin_city>
          <destination_city>ATL</destination_city>
          <stopover_code>1</stopover_code>
          <fare_basis_code>0</fare_basis_code>
          <flight_number/>
        </leg>
      </legs>
    </travel>
  </payment_transaction>'
```

**Non Travel Authorize (3D) and Travel Capture**

In this scenario the Authorize request doesn't contain the travel data. Valid travel data is submitted in the Capture transaction request. The travel data that will be used for the transaction processing is the data submitted with the Capture.

Example Non Travel Authorize

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Authorize');
    $request = $genesis->request();

    $request
        ->setTransactionId('1915c52b28ff29edb12bc617ac102a03')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('4200000000000000')
        ->setCvv('834')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.AuthorizeRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    AuthorizeRequest request = new AuthorizeRequest();

    request.setTransactionId("1915c52b28ff29edb12bc617ac102a03");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("4200000000000000");
    request.setCvv("834");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.authorize(
  {
    "transaction_id": "1915c52b28ff29edb12bc617ac102a03",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "4200000000000000",
    "cvv": "834",
    "expiration_month": "12",
    "expiration_year": 2021
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>authorize</transaction_type>
    <transaction_id>1915c52b28ff29edb12bc617ac102a03</transaction_id>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>100</amount>
    <currency>USD</currency>
    <card_holder>Travis Pastrana</card_holder>
    <card_number>4200000000000000</card_number>
    <cvv>834</cvv>
    <expiration_month>12</expiration_month>
    <expiration_year>2021</expiration_year>
  </payment_transaction>'
```

Example Travel Capture

Request

```
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Capture');
    $request = $genesis->request();

    $request
        ->setTransactionId('7fcc6097153beeec1dd1e71f643198d8')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setReferenceId('1f2ee425e7c3159c60743098071771eb')
        ->setTravel('{"ticket"=>{"ticket_number"=>12345678123456, "passenger_name"=>"Passenger 01", "customer_code"=>"", "restricted_ticket_indicator"=>0, "agency_name"=>"New Agency", "agency_code"=>"AGN001"},

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.CaptureRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    CaptureRequest request = new CaptureRequest();

    request.setTransactionId("7fcc6097153beeec1dd1e71f643198d8");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setReferenceId("1f2ee425e7c3159c60743098071771eb");
    request.setTravel("{"ticket"=>{"ticket_number"=>12345678123456, "passenger_name"=>"Passenger 01", "customer_code"=>"", "restricted_ticket_indicator"=>0, "agency_name"=>"New Agency", "agency_code"=>"AGN001"

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.capture(
  {
    "transaction_id": "7fcc6097153beeec1dd1e71f643198d8",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "reference_id": "1f2ee425e7c3159c60743098071771eb",
    "travel": {
      "ticket": {
        "ticket_number": 12345678123456,
        "passenger_name": "Passenger 01",
        "customer_code": "",
        "restricted_ticket_indicator": 0,
        "agency_name": "New Agency",
        "agency_code": "AGN001"
      },
      "legs": [
        {
          "departure_date": "2017-03-10",
          "carrier_code": "VX",
          "service_class": "J",
          "origin_city": "DUB",
          "destination_city": "ATL",
          "stopover_code": 1,
          "fare_basis_code": 0,
          "flight_number": ""
        }
      ]
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>capture</transaction_type>
    <transaction_id>7fcc6097153beeec1dd1e71f643198d8</transaction_id>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>100</amount>
    <currency>USD</currency>
    <reference_id>1f2ee425e7c3159c60743098071771eb</reference_id>
    <travel>
      <ticket>
        <ticket_number>12345678123456</ticket_number>
        <passenger_name>Passenger 01</passenger_name>
        <customer_code/>
        <restricted_ticket_indicator>0</restricted_ticket_indicator>
        <agency_name>New Agency</agency_name>
        <agency_code>AGN001</agency_code>
      </ticket>
      <legs>
        <leg>
          <departure_date>2017-03-10</departure_date>
          <carrier_code>VX</carrier_code>
          <service_class>J</service_class>
          <origin_city>DUB</origin_city>
          <destination_city>ATL</destination_city>
          <stopover_code>1</stopover_code>
          <fare_basis_code>0</fare_basis_code>
          <flight_number/>
        </leg>
      </legs>
    </travel>
  </payment_transaction>'
```

**Travel Authorize (3D) and Non Travel Capture**

In this scenario the Authorize request contains valid travel data. Travel data isn't submitted in the Capture transaction request. In this case the Capture transaction will inherit the travel data from the Authorize transaction. The travel data that will be used for the transaction processing is the data submitted with the Authorize.

Example Travel Authorize

Request

```
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Authorize');
    $request = $genesis->request();

    $request
        ->setTransactionId('1915c52b28ff29edb12bc617ac102a03')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('4200000000000000')
        ->setCvv('834')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setTravel('{"ticket"=>{"ticket_number"=>12345678123456, "passenger_name"=>"Passenger 01", "customer_code"=>123, "restricted_ticket_indicator"=>0, "agency_name"=>"Agency", "agency_code"=>"AG001"}, "le

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.AuthorizeRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    AuthorizeRequest request = new AuthorizeRequest();

    request.setTransactionId("1915c52b28ff29edb12bc617ac102a03");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("4200000000000000");
    request.setCvv("834");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setTravel("{"ticket"=>{"ticket_number"=>12345678123456, "passenger_name"=>"Passenger 01", "customer_code"=>123, "restricted_ticket_indicator"=>0, "agency_name"=>"Agency", "agency_code"=>"AG001"}, "

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.authorize(
  {
    "transaction_id": "1915c52b28ff29edb12bc617ac102a03",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "4200000000000000",
    "cvv": "834",
    "expiration_month": "12",
    "expiration_year": 2021,
    "travel": {
      "ticket": {
        "ticket_number": 12345678123456,
        "passenger_name": "Passenger 01",
        "customer_code": 123,
        "restricted_ticket_indicator": 0,
        "agency_name": "Agency",
        "agency_code": "AG001"
      },
      "legs": [
        {
          "departure_date": "2017-03-10",
          "carrier_code": "VX",
          "service_class": "J",
          "origin_city": "DUB",
          "destination_city": "ATL",
          "stopover_code": 1,
          "fare_basis_code": 0,
          "flight_number": ""
        }
      ]
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>authorize</transaction_type>
    <transaction_id>1915c52b28ff29edb12bc617ac102a03</transaction_id>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>100</amount>
    <currency>USD</currency>
    <card_holder>Travis Pastrana</card_holder>
    <card_number>4200000000000000</card_number>
    <cvv>834</cvv>
    <expiration_month>12</expiration_month>
    <expiration_year>2021</expiration_year>
    <travel>
      <ticket>
        <ticket_number>12345678123456</ticket_number>
        <passenger_name>Passenger 01</passenger_name>
        <customer_code>123</customer_code>
        <restricted_ticket_indicator>0</restricted_ticket_indicator>
        <agency_name>Agency</agency_name>
        <agency_code>AG001</agency_code>
      </ticket>
      <legs>
        <leg>
          <departure_date>2017-03-10</departure_date>
          <carrier_code>VX</carrier_code>
          <service_class>J</service_class>
          <origin_city>DUB</origin_city>
          <destination_city>ATL</destination_city>
          <stopover_code>1</stopover_code>
          <fare_basis_code>0</fare_basis_code>
          <flight_number/>
        </leg>
      </legs>
    </travel>
  </payment_transaction>'
```

Example Non Travel Capture

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Capture');
    $request = $genesis->request();

    $request
        ->setTransactionId('7fcc6097153beeec1dd1e71f643198d8')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setReferenceId('1f2ee425e7c3159c60743098071771eb');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.CaptureRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    CaptureRequest request = new CaptureRequest();

    request.setTransactionId("7fcc6097153beeec1dd1e71f643198d8");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setReferenceId("1f2ee425e7c3159c60743098071771eb");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.capture(
  {
    "transaction_id": "7fcc6097153beeec1dd1e71f643198d8",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "reference_id": "1f2ee425e7c3159c60743098071771eb"
  }
).send()
  .then(success)
  .catch(failure);
```

```curl
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>capture</transaction_type>
    <transaction_id>7fcc6097153beeec1dd1e71f643198d8</transaction_id>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>100</amount>
    <currency>USD</currency>
    <reference_id>1f2ee425e7c3159c60743098071771eb</reference_id>
  </payment_transaction>'
```

**Partial Travel Authorize (3D) and Partial Travel Capture**

In this scenario the Authorize request contains only part of the travel data. The Capture transaction request contains also part of the travel data. The travel data from the Authorize will be merged with the Capture travel data. The Capture travel data will complete/override the travel fields in the Authorize. This merged data will be stored as Capture travel data. If the merged data is valid travel data then the transaction will be processed as travel using the travel data stored in the Capture. Otherwise the transaction will be processed as a regular Capture transaction.

Example Partial Authorize

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Cards\Authorize');
    $request = $genesis->request();

    $request
        ->setTransactionId('1915c52b28ff29edb12bc617ac102a03')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setCardHolder('Travis Pastrana')
        ->setCardNumber('4200000000000000')
        ->setCvv('834')
        ->setExpirationMonth('12')
        ->setExpirationYear('2021')
        ->setTravel('{"ticket"=>{}, "legs"=>[{"departure_date"=>"2017-03-10", "carrier_code"=>"VX", "service_class"=>"J", "origin_city"=>"DUB", "destination_city"=>"ATL", "stopover_code"=>1, "fare_basis_code"=

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.card.AuthorizeRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    AuthorizeRequest request = new AuthorizeRequest();

    request.setTransactionId("1915c52b28ff29edb12bc617ac102a03");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setCardHolder("Travis Pastrana");
    request.setCardNumber("4200000000000000");
    request.setCvv("834");
    request.setExpirationMonth("12");
    request.setExpirationYear("2021");
    request.setTravel("{"ticket"=>{}, "legs"=>[{"departure_date"=>"2017-03-10", "carrier_code"=>"VX", "service_class"=>"J", "origin_city"=>"DUB", "destination_city"=>"ATL", "stopover_code"=>1, "fare_basis_code

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.authorize(
  {
    "transaction_id": "1915c52b28ff29edb12bc617ac102a03",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "card_holder": "Travis Pastrana",
    "card_number": "4200000000000000",
    "cvv": "834",
    "expiration_month": "12",
    "expiration_year": 2021,
    "travel": {
      "ticket": {
      },
      "legs": [
        {
          "departure_date": "2017-03-10",
          "carrier_code": "VX",
          "service_class": "J",
          "origin_city": "DUB",
          "destination_city": "ATL",
          "stopover_code": 1,
          "fare_basis_code": 0,
          "flight_number": ""
        }
      ]
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>authorize</transaction_type>
    <transaction_id>1915c52b28ff29edb12bc617ac102a03</transaction_id>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>100</amount>
    <currency>USD</currency>
    <card_holder>Travis Pastrana</card_holder>
    <card_number>4200000000000000</card_number>
    <cvv>834</cvv>
    <expiration_month>12</expiration_month>
    <expiration_year>2021</expiration_year>
    <travel>
      <ticket/>
      <legs>
        <leg>
          <departure_date>2017-03-10</departure_date>
          <carrier_code>VX</carrier_code>
          <service_class>J</service_class>
          <origin_city>DUB</origin_city>
          <destination_city>ATL</destination_city>
          <stopover_code>1</stopover_code>
          <fare_basis_code>0</fare_basis_code>
          <flight_number/>
        </leg>
      </legs>
    </travel>
  </payment_transaction>'
```

Example Partial Capture

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Capture');
    $request = $genesis->request();

    $request
        ->setTransactionId('7fcc6097153beeec1dd1e71f643198d8')
        ->setRemoteIp('245.253.2.12')
        ->setAmount('100')
        ->setCurrency('USD')
        ->setReferenceId('1f2ee425e7c3159c60743098071771eb')
        ->setTravel('{"ticket"=>{"ticket_number"=>12345678123456, "passenger_name"=>"Passenger 01", "customer_code"=>"", "restricted_ticket_indicator"=>0, "agency_name"=>"New Agency", "agency_code"=>"AGN001"},

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.CaptureRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;
import java.math.BigDecimal;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    CaptureRequest request = new CaptureRequest();

    request.setTransactionId("7fcc6097153beeec1dd1e71f643198d8");
    request.setRemoteIp("245.253.2.12");
    request.setAmount(new BigDecimal(100));
    request.setCurrency("USD");
    request.setReferenceId("1f2ee425e7c3159c60743098071771eb");
    request.setTravel("{"ticket"=>{"ticket_number"=>12345678123456, "passenger_name"=>"Passenger 01", "customer_code"=>"", "restricted_ticket_indicator"=>0, "agency_name"=>"New Agency", "agency_code"=>"AGN001"

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.capture(
  {
    "transaction_id": "7fcc6097153beeec1dd1e71f643198d8",
    "remote_ip": "245.253.2.12",
    "amount": "100",
    "currency": "USD",
    "reference_id": "1f2ee425e7c3159c60743098071771eb",
    "travel": {
      "ticket": {
        "ticket_number": 12345678123456,
        "passenger_name": "Passenger 01",
        "customer_code": "",
        "restricted_ticket_indicator": 0,
        "agency_name": "New Agency",
        "agency_code": "AGN001"
      },
      "legs": [

      ]
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
   -X POST \
   -H "Content-Type: text/xml" \
   -d '
  <?xml version="1.0" encoding="UTF-8"?>
  <payment_transaction>
    <transaction_type>capture</transaction_type>
    <transaction_id>7fcc6097153beeec1dd1e71f643198d8</transaction_id>
    <remote_ip>245.253.2.12</remote_ip>
    <amount>100</amount>
    <currency>USD</currency>
    <reference_id>1f2ee425e7c3159c60743098071771eb</reference_id>
    <travel>
      <ticket>
        <ticket_number>12345678123456</ticket_number>
        <passenger_name>Passenger 01</passenger_name>
        <customer_code/>
        <restricted_ticket_indicator>0</restricted_ticket_indicator>
        <agency_name>New Agency</agency_name>
        <agency_code>AGN001</agency_code>
      </ticket>
      <legs/>
    </travel>
  </payment_transaction>'
```

**VISA REFUND**

The additional visa refund request parameters are applicable only when the reference transaction isAirline Itinerary Data (AID) or Ancillary Charges.

Visa Refund

Request

```php
<?php

// Load the pre-configured ini file...
\Genesis\Config::loadSettings('/path/to/config.ini');

try {
    $genesis = new Genesis('Financial\Refund');
    $request = $genesis->request();

    $request
        ->set...('')
        ->setTravel('{"ticket"=>{"credit_reason_indicator_1"=>"C", "credit_reason_indicator_2"=>"A", "ticket_change_indicator"=>"B"}}');

    $genesis->execute();
    $response = $genesis->response()->getResponseObject();
} catch (\Genesis\Exceptions\ErrorAPI $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\InvalidArgument $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\ErrorParameter $e) {
    $response = $e->getMessage();
} catch (\Genesis\Exceptions\Exception $e) {
    $response = $e->getMessage();
}
```

```java
import com.emerchantpay.gateway.GenesisClient;
import com.emerchantpay.gateway.api.TransactionResult;
import com.emerchantpay.gateway.api.constants.Endpoints;
import com.emerchantpay.gateway.api.constants.Environments;
import com.emerchantpay.gateway.api.requests.financial.RefundRequest;
import com.emerchantpay.gateway.model.Transaction;
import com.emerchantpay.gateway.util.Configuration;
import com.emerchantpay.gateway.util.Country;
import com.emerchantpay.gateway.util.Currency;

public class GenesisExample {
  public static void main()  {

    // Create configuration
    Configuration configuration = new Configuration(Environments.STAGING, Endpoints.EMERCHANTPAY);

    configuration.setUsername("SET_YOUR_USERNAME");
    configuration.setPassword("SET_YOUR_PASSWORD");
    configuration.setToken("SET_YOUR_TOKEN");

    RefundRequest request = new RefundRequest();

    request.set...("");
    request.setTravel("{"ticket"=>{"credit_reason_indicator_1"=>"C", "credit_reason_indicator_2"=>"A", "ticket_change_indicator"=>"B"}}");

    GenesisClient client = new GenesisClient(configuration, request);
    client.execute();

    // Parse Payment result
    System.out.println(client.getResponse());

  }
}
```

```javascript
var genesis = require('genesis.js/lib/genesis.js');
var transaction = new genesis.transaction();

var failure = function(reason) {
  return console.log(reason);
};

var success = function(data) {
  return console.log(data);
};

transaction.refund(
  {
    "...": "",
    "travel": {
      "ticket": {
        "credit_reason_indicator_1": "C",
        "credit_reason_indicator_2": "A",
        "ticket_change_indicator": "B"
      }
    }
  }
).send()
  .then(success)
  .catch(failure);
```

```
curl https://username:password@staging.gate.e-comprocessing.net/process/TERMINAL-TOKEN \
  -X POST \
  -H "Content-Type: text/xml" \
  -d '
 <?xml version="1.0" encoding="UTF-8"?>
 <payment_transaction>
   <.../>
   <travel>
     <ticket>
       <credit_reason_indicator_1>C</credit_reason_indicator_1>
       <credit_reason_indicator_2>A</credit_reason_indicator_2>
       <ticket_change_indicator>B</ticket_change_indicator>
     </ticket>
   </travel>
 </payment_transaction>'
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| **travel** | optional | | |
| **ticket** | optional | | |
| credit_reason_indicator_1 | optional | String(1) | This field indicates the reason for a credit to the cardholder. Allowed values: A, B, P, O |
| credit_reason_indicator_2 | optional | String(1) | This field indicates the reason for a credit to the cardholder. Allowed values: A, B, P, O |
| ticket_change_indicator | optional | String(1) | This field will contain either a space or a code to indicate why a ticket was changed. Allowed values: C, N |

`required* = conditionally required`

---

# Allowed Values

**CAR RENTAL CLASSES**

| Value | Description |
|---|---|
| 1 | Mini |
| 2 | Subcompact |
| 3 | Economy |
| 4 | Compact |
| 5 | Midsize |
| 6 | Intermediate |
| 7 | Standard |
| 8 | Full size |
| 9 | Luxury |
| 10 | Premium |
| 11 | Minivan |
| 12 | 12 passenger van |
| 13 | Moving van |
| 14 | 15 passenger van |
| 15 | Cargo van |
| 16 | 12 foot truck |
| 17 | 20 foot truck |
| 18 | 24 foot truck |
| 19 | 26 foot truck |
| 20 | Moped |
| 21 | Stretch limousine |
| 22 | Regular limousine |
| 23 | Unique limousine |

| Value | Description |
|---|---|
| 24 | Exotic limousine |
| 25 | Small/medium truck |
| 26 | Large truck |
| 27 | Small SUV |
| 28 | Medium SUV |
| 29 | Large SUV |
| 30 | Exotic SUV |
| 9999 | Miscellaneous |

**CHARGE TYPES**

| Value | Description |
|---|---|
| BF | Bundled Service |
| BG | Baggage Fee |
| CF | Change Fee |
| CG | Cargo |
| CO | Carbon Offset |
| FF | Frequent Flyer |
| GF | Gift Card |
| GT | Ground Transport |
| IE | In-flight Entertainment |
| LG | Lounge |
| MD | Medical |
| ML | Meal / Beverage |
| OT | Other |
| PA | Passenger Assist Fee |
| PT | Pets |
| SA | Seat Fees |
| SB | Standby |
| SF | Service Fee |
| ST | Store |
| TS | Travel Service |
| UN | Unaccompanied Travel |
| UP | Upgrades |
| WI | Wi-Fi |
| MISC | Miscellaneous Airline Charges |

**CAR RENTAL EXTRA CHARGES**

| Value | Description |
|---|---|
| 1 | Gas |
| 2 | Extra Mileage |
| 3 | Late Return |
| 4 | 1 Way Ser Fee |
| 5 | Parking Violation |

**HOTEL RENTAL EXTRA CHARGES**

| Value | Description |
|---|---|

| Value | Description |
|---|---|
| 2 | Restaurant |
| 3 | Gift Shop |
| 4 | Mini Bar |
| 5 | Telephone |
| 6 | Laundry |
| 7 | Other |

**TICKET CHANGE INDICATORS**

| Value | Description |
|---|---|
| C | Change to existing Ticket |
| N | New ticket |

**CREDIT REASON INDICATORS**

| Value | Description |
|---|---|
| A | Passenger Transport Ancillary Cancellation |
| B | Travel Ticket and Passenger Transport |
| P | Partial Refund of Travel Ticket |
| O | Other |

# Genesis SCA Services

## General Info

Genesis SCA(Strong Customer Authentication) Services provides the ability to check if a transaction is in the scope of SCA. The API is synchronous and is based on RESTFul practices. Be sure to set `Content-type: application/json` in your headers.

To interact with the SCA API, you need to provide login credentials using standard HTTP Basic Authentication. (Credentials can be found in your Admin interface.)

---

## SCA Checker

This call is used to check if SCA is required

`POST /v1/sca/checker/:terminal_token`

Request

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
curl https://username:password@staging.gate.e-comprocessing.net/v1/sca/checker/:terminal_token \
  -X POST \
  -H "Content-Type: application/json" \
  -d '
 {
   "card_number": "4200000000000000",
   "moto": false,
   "mit": false,
   "recurring": false,
   "transaction_amount": 350000,
   "transaction_currency": "EUR",
   "transaction_exemption": ""
 }'
```

Successful Response

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
This request is not implemented yet
```

```
[
  {
    "sca_required": "no",
    "sca_result_reason": "Issuer out of the EEA",
    "exemption": null
  }
]
```

**Request Parameters**

| Parameter | Required | Format | Description |
|---|---|---|---|
| card_number | required | string(16) | Full card number or first 6 digits. |
| transaction_amount | required | number | Amount of transaction in minor currency unit. |
| transaction_currency | required | string(3) | Transaction currency |
| moto | optional | boolean | Signifies whether a MOTO (mail order telephone order) transaction is performed. |
| mit | optional | boolean | Signifies whether a MIT (merchant initiated transaction) is performed. |
| recurring | optional | boolean | Signifies whether a Recurring Sale transaction is performed. |
| transaction_exemption | optional | string(30) | Exemption |

`required* = conditionally required`

**Successful Response Parameters**

| Parameter | Type | Description |
|---|---|---|
| sca_required | string | Sca Required. Possible values are `yes`, `possible_exemption` or `no` |
| sca_result_reason | string | The reason for the returned SCA required |
| exemption | string | Detected exemption. Possible values are `low_value`, `low_risk`, `trusted_merchant`, `corporate_payment` or `delegated_authentication`. |