Paycryptos API documentation v.1.0

Table of contents

Change History

1.00	28 Aug 2020	Added new following sections: ● Table of contents ● Prerequisites - includes main required steps to prepare the client's system logic ● Deposit and Withdrawal algorithms ● CallBacks Sections rearrangement, erratum corrections, updated sections description Added Holistic Picture Addresses section removed

Introduction

This document describes the Paycryptos REST API and everything that is necessary to access it's resources. In order to use the API, you first need to have a verified Paycryptos account. If you did not receive your credentials yet, please contact support@paycryptos.com

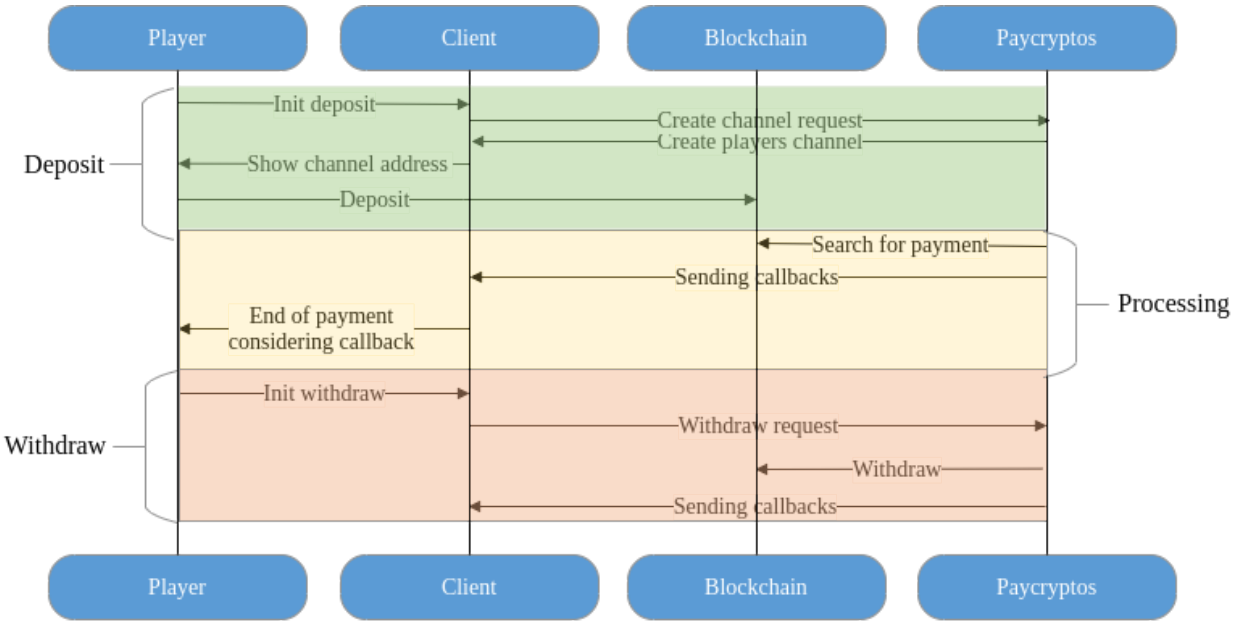
Prerequisites

- An account in PayCryptos needs to be set up by our customer service and credentials must be provided to you. You also need to set up IP whitelists on both sides.
- Required crypto currencies and fiat currencies are switched on for your account in PayCryptos
- Each user (meaning, player) is designated in a particular currency. It can be a crypto currency, eg. Bitcoin account (BTC), Ethereum account (ETH), etc. Or it can be a fiat currency, for example EUR or USD.
- There is an on the fly conversion, that you can allow or disallow. For example, you can have EUR users access only to Bitcoin, USD users to Bitcoin and Ethereum, Bitcoin users only to Bitcoin (so, no on the fly conversion in that case, only native cryptocurrency), etc.
- You can also set up currency conversion margins for such on the fly conversions (so that buy and sell transactions use a slightly different exchange rate, just like a bank FX) or you can have 0% margins.
- If a user is created in EUR, you will receive both the user currency deposit amount and the original amount and the currency that was actually deposited.
- Conversion rates for crypto into EUR and EUR into other fiat currencies are kept up to date to ensure the amounts deposited / withdrawn will be correct.
- Each user will receive a unique wallet address (there is a slightly different logic in different crypto algorithms). Any payment to this wallet address will be considered as the payments from the user and if there is a pending (new) deposit request from the user, such payment will be considered as a payment by the user and a callback will be generated.
- The wallet addresses need to be prepared inside each wallet (there is a function in advanced wallets to do so) and then uploaded to PayCryptos for the usage. Those need to be replenished, as once they are used up, you won't be able to generate a unique wallet for users and an error will be sent to the owners.
- You also need to set up the number of confirmations for each crypto currency. The confirmation for the deposit and confirmation for the release of funds are two separate settings. Usually you want the user to receive the funds for playing as soon as possible, as even a single block chain confirmation can take hours if there is a congestion.
- Therefore we suggest you to keep track of each deposit status, as you will receive three callbacks for each successful deposit: xxxx, xxxx, xxxx. If you set up deposits to be credited without a safe number of confirmations, such deposits should not be allowed in activity that allows such deposits to be passed out of the system: eg. possibly sportsbook bets, withdrawals, poker and other p2p transactions should not be allowed. If you do not want to create this logic, you can put a higher number of transactions required to get a deposit

successfully credited to the account. But this will mean that the user will often wait for 30+ minutes to get the funds.

Holistic Picture

Player – user's web browser
Client – client system with own website
Paycryptos – SoftGamings financial system



Deposit Algorithm

1. Before accepting deposits, a channel needs to be created for a user(meaning a payer or a player).It will be created on a first request and all subsequent requests will just return info about a channel.
2. A user makes a deposit to the channel address.
3. Our blockchain monitoring system searches for deposits by channel addresses.
4. A callback for found deposit is described in the Callbacks section.

Withdrawal Algorithm

1. A user creates a withdrawal request, but he can cancel it before it will be processed.
2. A withdrawal request can be approved or declined.
3. A payout file will be generated from approved requests and payout will be made.

4. Withdrawal callbacks are in development right now. Upon completion, callbacks will be sent by the same channel URL as deposit callbacks but with different payment statuses and negative amounts.

Configuration

The base URL host for all API requests documented below is: https://paycryptos.com/api/v1
All API requests are performed over HTTPS and follow JSON API conventions. All data is sent and received as JSON with the content type application/json.

Request and Response Format

All API requests are performed over HTTPS and follow JSON API conventions. All data is sent and received as JSON with the content type application/json.

Authentication Headers

All requests to the API must include the following headers for authentication:

```
X-Cryptspay-Key: ...
X-Cryptspay-Nonce: ...
X-Cryptspay-Signature: ...
```

Using curl:

```
curl -H "X-Cryptspay-Key: ..." -H "X-Cryptspay-Nonce: ..." -H "X-Cryptspay-Signature: ..." ...
```

See Authentication section for a detailed description

GET Requests

It is recommended to set the "Accept" header as follows:

```
Accept: application/json
```

Using curl:

```
curl -H "Accept: application/json" ...
```

POST Requests

POST requests must send data in JSON format within a request body and have a header

```
Content-Type: application/json:
```

```
POST /api/v1/test
Content-Type: application/json
Accept: application/json

{
        "foobar": "some value",
        "foo": 12345,
        "bar": 0.75
}
```

Using curl:

```
curl -H "Content-Type: ..." -H "Accept: ..." -X POST \
     -d '{"foobar": "some value", "foo": 12345, "bar": 0.75}' ...
```

Responses

All API server responses are in JSON format with a Content-Type application/json:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
        "result": "OK"
}
```

Appropriate HTTP status codes are used both for successful processing and in case of errors.

Success Responses

Successful responses have one of the following HTTP status codes:

Situation	HTTP status code
Request was accepted, validated and processed	200 OK
Same as above, plus a resource was created as a result	201 Created

All success responses always have the following key:

Key	Type	Description
result	string	For success result is always OK

Error Responses

In case of any error a server responds with an appropriate HTTP status code and a JSON body:

```
HTTP/1.1 406 Forbidden
Content-Type: application/json
{
        "result": "FAIL",
        "message": "Request signature mismatch"
}
```

All error responses have the following format:

Key	Type	Description
result	string	For errors result is always FAIL
message	string	Human readable description of the error

Authentication

To use the Paycryptos API you must use an API token and an API Secret from your Paycryptos account. All requests to the API have to be authenticated using this token information in the headers.

Each valid authenticated request has to include the following HTTP headers:

X-Cryptspay-Key: ...
X-Cryptspay-Nonce: ...
X-Cryptspay-Signature: ...

Paycryptos API key: X-Cryptspay-Key

X-Cryptspay-Key is the API access key which you receive when you generate an API access token. It is a sequence of hex-digits represented as a string in uid format, randomly generated when an account is created.

API access keys are case-sensitive.

Example
X-Cryptspay-Key: bce83fb6-2577-4e56-9984-e8fa7661abba

Nonce: X-Cryptspay-Nonce

X-Cryptspay-Nonce is a 64-bit integer number, which you must generate for every request you make to the API. This nonce ("number used once") has to meet the following two requirements:

1. Nonce must be unique for every request you make with the same API access key, ever. If you make a request with the same API access key and nonce again, it will be rejected.
2. Every nonce that you generate for the request, has to be greater than any of the previous nonces that you used to make requests to the API. There is no way to reset the nonce value for a given API key but you can always just generate a new API key.

One way to generate nonces is to use the UNIX epoch timestamp of the request. Be sure, though, that you use enough precision: if you use only the seconds part of the timestamp and you send two requests to the API within the same second, one of them will be rejected. It is recommended that you use the UNIX epoch to microsecond resolution. The nonce value must be representable as an unsigned 64-bit integer therefore it has to be within the range [0..18446744073709551615]

Example
X-Cryptspay-Nonce: 1411754081462609

Paycryptos API secret:

The secret is a randomly generated 24 character long string from the character set:

'abcdefghijklmnopqrstuvwxyz!@#$%^&*()[]-+_.ABCDEFGHIJKLMNOP1234567890'.

Upon creation of a new API token, the secret is displayed in the web-interface.

Signature: X-Cryptspay-Signature

The signature is derived from the API secret, the nonce (X-Cryptspay-Nonce) and the request body. Its format is a hex-string representation of the result of the following hash calculation:

HMAC-SHA512(msg, k)

Where:

k - is your API secret as UTF-8 string

msg - is a UTF-8 string, constructed by string concatenation of **uri_path** + **nonce** + **SHA256(request_data)**

uri_path - is the path part of the request URI as UTF-8 string

nonce - is the nonce (X-Cryptspay-Nonce) of the request, converted to a UTF-8 string

SHA256(request_data) is the hex-encoded SHA256 digest of request_data , as a UTF-8 string, request_data is either the JSON encoded request body in case of POST requests, or the URL-encoded query in case of a GET request as specified in RFC3986

PHP example:

```
function signRequest($params) {
        $request_hash = hash('sha256', $params['request_data']);
        return hash_hmac(
                        'sha512',
                        implode('', [$params['path'], $params['nonce'], $request_hash]),
                        $params['secret']
        );
}
```

Testing (Ping)

GET /api/v1/ping/app

This request is intended to be used to test whether your application is configured properly and can access the Paycryptos API using GET requests

Request:
Any

Response:
On success, the API responds with **HTTP status 200 OK** and the following attributes:

Attribute	Data type	Description
result	string	OK

Errors:
On error, the API responds with standard error responses:

Situation	HTTP status code	MessageUnauthorized access
Invalid user api token	401 Unauthorized	Unauthorized access
Invalid request signature	406 Not Acceptable	Request signature mismatch

Example
Request:

```
GET /api/v1/ping/app HTTP/1.1
Accept: application/json
X-Cryptspay-Key: *****
X-Cryptspay-Nonce: *****
X-Cryptspay-Signature: *****
```

Response:

```
HTTP/1.1 200 OK
Content-Type: application/json
{
    "result": "OK"
}
```

POST /api/v1/ping/app

This request is intended to be used to test whether your application is configured properly and can access the Paycryptos API using POST requests

Request:
There are no required parameters, but you can pass any parameters to test whether your signature calculation is correct

Response:
On success, the API responds with **HTTP status 200 OK** and the following attributes:

Attribute	Data type	Description
result	string	OK

Errors:
On error, the API responds with standard error responses:

Situation	HTTP status code	MessageUnauthorized access
Invalid user api token	401 Unauthorized	Unauthorized access
Invalid request signature	406 Not Acceptable	Request signature mismatch

Example
Request:

```
POST /api/v1/ping/app HTTP/1.1
Content-Type: application/json
Accept: application/json
X-Cryptspay-Key: *****
X-Cryptspay-Nonce: *****
X-Cryptspay-Signature: *****
{
    "foo": "123.0",
    "bar": true
}
```

Response:

```
HTTP/1.1 200 OK
Content-Type: application/json
{
    "result": "OK"
}
```

Channels

Paycryptos channels are a way to receive and potentially convert cryptocurrency payments of variable amounts and without a set validity period. Cryptocurrency sent to a channel can be automatically converted into an associated payer currency at the current exchange rate at the moment of the payment and such conversion never changes after that. Channels have a fixed payment address and support callbacks. Channels also provide a way of tracking individual payments made to them. A list of transactions is recorded for each channel and returned upon querying a channel's transactions. Possible use cases for Paycryptos channels are accepting donations on a website or just keeping track of payments to a certain address.

POST /api/v1/channel/create

If a channel doesn't exist this request will create it and mark one free address as busy for this channel, otherwise a request will return channel info by requested wallet and payer identifier.

Request:
You must pass these required attributes:

Attribute	Data type	Description
external_id	string(255)	Payer identifier
external_name	string(255)	Payer name
wallet	string(36)	Wallet api key in UUID format
currency	string(3)	ISO 4217 code of the payer currency
callback_url	string(255)	Valid URL that is called when a channel status is updated
success_url	string(255)	Valid URL to redirect a user after a successful payment
cancel_url	string(255)	Valid URL to redirect a user after a failed payment

Response:
On success, the API responds with **HTTP status 200 OK** (if a channel already exists) or **HTTP status 201 Created** (if a channel doesn’t exist). A response has the following attributes in any of these cases:

Attribute	Data type	Description
id	string(36)	Unique generated channel identifier in UUID format
channel_url	string	Unique URL of the channel payment screen on Paycryptos
address	string(255)	Crypto address associated with this channel
currency	string(3)	ISO 4217 code of the user currency

Errors:
On error, the API responds with standard error responses and with some specific to this request:

Situation	HTTP status code	Message
Requested currency is not supported for current user	404 Not Found	Currency not supported by system
Requested currency is supported, but now is disabled	404 Not Found	Currency temporarily disabled
Wallet not found by requested wallet key or this key is incorrect	404 Not Found	Wallet not found
Requested currency is not supported for the wallet found	404 Not Found	Unsupported wallet currency
Requested wallet does't contain free crypto addresses	404 Not Found	Not found any free addresses for wallet
Detected concurrent request	409 Conflict	Concurrent request detected
Error occurred on address reservation for channel	500 Internal Server Error	Error occurred on address reservation for channel
Error occurred on create channel	500 Internal Server Error	Error occurred on create channel
System error occurred	500 Internal Server Error	System error occurred

Example

Request:

```
POST /api/v1/channel/create HTTP/1.1
Content-Type: application/json
Accept: application/json
X-Cryptspay-Key: *****
X-Cryptspay-Nonce: *****
X-Cryptspay-Signature: *****
{
    "external_id": "201879",
    "external_name": "164_275",
    "wallet": "8fbe129b-99ae-4402-a1dd-788bdc641c15",
    "currency": "EUR",
    "callback_url": "https://example.com/callback",
    "success_url": "https://example.com/success",
    "cancel_url": "https://example.com/cancel"
}
```

Response:

```
HTTP/1.1 201 Created  or  HTTP/1.1 200 OK
Content-Type: application/json
{
    "id" : "96bbf7e7-79e0-4d08-8555-3c39998e86d0",
    "channel_url" :
"https://paycryptos.com/channels/96bbf7e7-79e0-4d08-8555-3c39998e86d0",
    "address" : "4tYpFBhpzV7UJ3tskv7Sv9giDVWemkeHF72",
    "currency" : "EUR"
}
```

GET /api/v1/info/{uid}

Get info about a channel by channel uid

Request:

You must pass these required attributes:

Attribute	Data type	Description
uid	string(36)	Unique identifier of a channel in UUID format

Response:

On success API responds with **HTTP status 200 OK** and the following attributes:

Attribute	Data type	Description
id	string(36)	Unique generated channel identifier in UUID format
channel_url	string	Unique URL of the channel payment screen on Paycryptos
address	string(255)	Crypto address associated with this channel
currency	string(3)	ISO 4217 code of the channel currency

Errors:
On error, the API responds with standard error responses and with some specific to this request:

Sutiation	HTTP status code	Message
Request without required uid attribute	406 Not Acceptable	Channel identificator not set
Channel wasn't found by requested uid or this uid is incorrect	404 Not Found	Channel not found or doesn’t belong to user
System error occurred	500 Internal Server Error	System error occurred

Example
Request:

```
GET api/v1/info/96bbf7e7-79e0-4d08-8555-3c39998e86d0 HTTP/1.1
Accept: application/json
X-Cryptspay-Key: *****
X-Cryptspay-Nonce: *****
X-Cryptspay-Signature: *****
```

Response:

```
HTTP/1.1 200 OK
Content-Type: application/json
{
        "id" : "96bbf7e7-79e0-4d08-8555-3c39998e86d0",
        "channel_url" :
"https://paycryptos.com/channels/96bbf7e7-79e0-4d08-8555-3c39998e86d0",
        "address" : "4tYpFBhpzV7UJ3tskv7Sv9giDVWemkeHF72",
        "currency" : "EUR"
}
```

Withdraws

POST /api/v1/withdraws/request/{uid}

Create withdrawal request by channel uid

Request:
You must pass these required request attributes:

Attribute	Data type	Description
uid	string(36)	Unique identifier of the channel in UUID format

You must pass these required request parameters:

Attribute	Data type	Description
address	string(255)	Crypto address to withdraw
amount	string	Withdrawal amount in channel currency
external_txid	string(255)	External system transaction identifier

Response:
On success, the API responds with **HTTP status 200 OK** and json array of elements, each element of this array has the following attributes:

Attribute	Data type	Description
result	string	OK or FAIL
amount	string	Wallet currency amount
fiat_ex_rate	string	Fiat currency backrate
crypto_ex_rate	string	Crypto currency backrate
id	int	Withdraw request identifier on our side

Errors:
On error, the API responds with standard error responses and with one specific to this request:

Situation	HTTP status code	Message
Withdrawal api not enabled for user	401 Unauthorized	You can“t use withdrawal api
Request without required params	406 Not Acceptable	Request not acceptable

Invalid channel uid	404 Not Found	Unknown channel
Withdrawal request already exists	500 Internal Server Error	Withdrawal request already exists
Invalid withdrawal address	406 Not Acceptable	Withdrawal address is invalid
Insufficient wallet balance	406 Not Acceptable	Insufficient wallet balance

Example
Request:

POST /api/v1/withdraws/request/96bbf7e7-79e0-4d08-8555-3c39998e86d0 HTTP/1.1
Content-Type: application/json
Accept: application/json
X-Cryptspay-Key: *****
X-Cryptspay-Nonce: *****
X-Cryptspay-Signature: *****

Response:

HTTP/1.1 200 OK
Content-Type: application/json
{

"result" => "OK"
"id" : 12223,
"amount" : "0.00999834"
"crypto_ex_rate" : "0.0002323",
"fiat_ex_rate":"0.00334334"

}

POST /api/v1/withdraws/cancel/{id}

Canceling a withdrawal request based on an external system identifier. It works if the request has not been processed

Request:
You must pass these required request attributes:

Attribute	Data type	Description
id	string(255)	External system withdrawal request identifier

Response:
On success, the API responds with **HTTP status 200 OK** and json array of elements, each element of this array has the following attributes:

Attribute	Data type	Description
result	string	OK or FAIL
id	string	External system withdrawal request identifier
status	string	cancelled

Errors:
On error, the API responds with standard error responses and with one specific to this request:

Situation	HTTP status code	Message
Withdrawal api not enabled for user	401 Unauthorized	You can’t use withdrawal api
Invalid external request id	404 Not Found	Withdrawal request not found
Withdrawal request is processing	304 Not Modified	Withdrawal request is already processing

Example
Request:

```
POST /api/v1/withdraws/cancel/12223 HTTP/1.1
Content-Type: application/json
Accept: application/json
X-Cryptspay-Key: *****
X-Cryptspay-Nonce: *****
X-Cryptspay-Signature: *****
```

Response:

```
HTTP/1.1 200 OK
Content-Type: application/json
{

        "result" => "OK"
        "id" : 12223,
        "status" : "cancelled"

}
```

Currencies

GET /api/v1/currency/rate/{pair}

Get exchange rate for a currency pair

Request:
You must pass these required attributes:

Attribute	Data type	Description
pair	string(7)	Pair of different currencies, example BTC_USD

Response:
On success, the API responds with **HTTP status 200 OK** and the following attributes:

Attribute	Data type	Description
base	string	Base fiat currency for system, example USD
rate	float	Exchange rate
crypto_rate	float	Crypto rate
fiat_rate	float	Fiat rate

Errors:
On error, the API responds with standard error responses and with some specific to this request:

Situation	HTTP status code	Message
Requested currency pair is invalid	404 Not Found	Invalid currency pair
One or more requested currencies not found	404 Not Found	Currency not found
Get the exchange rate from crypto to crypto is not supported	406 Not Acceptable	Not supported rate pair crypto_crypto
Get the exchange rate from fiat to fiat is not supported	406 Not Acceptable	Not supported rate pair fiat_fiat

Example
Request:

```
GET /api/v1/currency/rate/BTC_EUR HTTP/1.1
Accept: application/json
X-Cryptspay-Key: *****
X-Cryptspay-Nonce: *****
X-Cryptspay-Signature: *****
```

Response:

HTTP/1.1 200 OK
Content-Type: application/json


```
{
    "base" : "USD",
    "rate" : "1"
    "crypto_rate" : "1"
    "fiat_rate" : "1"
}
```

Callbacks

Paycryptos sends a callback three times for each payment to the channel's callback url. Each callback corresponds to three states of a payment: **new**, **confirmed**, and **unblocked**.

new - the payment's state, when it is found in the mempool or blockchain, and it may be not included in the block;

confirmed - the payment's state, when it has reached the minimum required number of confirmations set in the wallet settings for deposits confirmations;

unblocked - the payment's state, when it has reached the maximum required number of confirmations set in the wallet settings for withdrawal confirmations;

Each callback contains signature, you can check it

HMAC-SHA512(msg, k)

Where:

k - is your API secret as UTF-8 string

msg - is a UTF-8 string, constructed by string concatenation of **callback_id** + **nonce** + **SHA256(request_data)**

callback_id - is the callback identificator as UTF-8 string

nonce - is the nonce (X-Cryptspay-Nonce) of the request, converted to a UTF-8 string

SHA256(request_data) - is the hex-encoded SHA256 digest of request_data , as a UTF-8 string, request_data is either the JSON encoded request body in case of POST requests, or the URL-encoded query in case of a GET request as specified in RFC3986

Example

Request:

```
POST https://apiprod.example.org/Paycryptos/Callback
Content-Type: application/json
Accept: application/json
X-Cryptspay-Key: *****
X-Cryptspay-Nonce: *****
X-Cryptspay-Callback: 196
X-Cryptspay-Signature: *****
{
    "status":"new",
    "amount":"0.10000000",
    "receiver":
    {
        "amount":"3.79894171",
        "currency":"EUR",
        "crypto_ex_rate":"43.42000000",
        "fiat_ex_rate":"0.87492900"
    },
```

"txid":"bfee8b15f6f0391f8242168df6bb9605d45312ae9e2cbc5204ea167e87f2f2ca",
"address":"QYMc9wrQunpJ7sMCQrpL3za4CRt7q1QR8C",
"channel_id":"ac1c9040-c498-45f1-8b71-65411ecbd15c",
"currency":"LTC",
"id":"196"
}

Response:

HTTP/1.1 200 OK
Content-Type: application/json

{
"result" : "OK",
}

Success Responses

Successful responses have one of the following HTTP status codes:

Situation	HTTP status code
Request was accepted, validated and processed	200 OK
Same as above, plus a resource was created as a result	201 Created

All success responses always have the following key:

Key	Type	Description
result	string	For success result always is OK

Error Responses

In case of any error a merchant server responds with an appropriate HTTP status code and a JSON body:

```
HTTP/1.1 200
Content-Type: application/json
{
        "result": "FAIL",
        "message": "Can't create payment by some reason"
}
```

All error responses have the following format:

Key	Type	Description
result	string	For errors result is always FAIL
message	string	Human readable description of an error