XMR.to API Documentation

Release 3.0

XMR.to

Nov 11, 2020

CONTENTS

This is the API documentation for XMR.to.

XMR.to - Pay the world with Monero.

Follow us on Twitter if you want to get updates about XMR.to: https://twitter.com/xmr_to

Note: The current version of the API is 3.

CHAPTER

ONE

CONTENTS

1.1 Introduction

XMR.to - Pay the world with Monero. Monero is a secure, private, untraceable currency that is open-source and freely available to all. XMR.to converts moneroj (XMR) provided by the user to bitcoins (BTC), which are sent to a given bitcoin address.

XMR.to provides an API that enables developers to use the service in programs, apps, scripts etc. This API therefore allows developers to integrate an option to pay any bitcoin address in their product, for example, in a Monero wallet service. The API is a REST-like API using *JSON* as data format. It does not require authentication. This document describes this API and gives examples of its usage.

Warning: You agree to XMR.to's Terms of Service and API Terms by accessing this API in any way or form. Read the Terms of Service here: https://xmr.to/terms-of-service and the API terms here: https://xmr.to/api-terms

1.1.1 Overview

In order to use XMR.to, a user must first create an order over a certain amount of bitcoins (BTC). Once the order has been created, XMR.to provides the user with the details for the payment in moneroj (XMR). Once the user has fully paid using Monero, XMR.to will create a bitcoin transaction over the given amount to the given bitcoin address.

When using the API, the general flow of events is similar:

- get current order parameters to see if XMR.to is available and to fetch current price, order limits, etc…
- create a new order by supplying bitcoin destination address and mount
- check order status to get payment information
- pay order
- continue to repeatedly check order status for processing progress

1.1.2 Naming conventions

API base URL

The base URL of the API is `https://xmr.to/api/`, followed by the version identifier (currently `v3`), followed by the conversion direction (currently only `xmr2btc`). Therefore, the complete API base URL is `https://xmr.to/api/v3/xmr2btc/`.

Note: The current version of the API is 3.

API endpoint names

API endpoints are always named `<noun>_<verb>`. For example, the endpoint to create a new order is called `/order_create/`. For example, for API version 3 the complete URL would be `https://xmr.to/api/v3/xmr2btc/order_create/`.

Field names and values

Fields are named using lower-case nouns separated by underscores. Values are given in their default format. For example, the field `btc_amount` gives the amount of an order in bitcoins rather than satoshis. Data types are not indicated in field names.

1.1.3 Data format and errors

Responses

API responses are always *JSON*-formatted data.

API errors

On success, the API returns the requested data in *JSON* format and *HTTP* return code `200`.

On failure, the API returns an error message formatted in *JSON* and a *HTTP* error code. The error message is formatted as follows:

```
{
    "error": "XMRTO-ERROR-<number>"
    "error_msg": "<error_message_as_string>"
    "error_msg_display": "<error_message_to_display_as_string>"
}
```

The `error` number is unique per message across all endpoints and can be used to reliably identify errors when they occur. The `error_msg` is a human-readable description of the error and should not be used by a script or program that uses the API. The same applies to `error_msg_display`. * `error_msg` - aimed at technical users * `error_msg_display` - aimed at regular non-technical users

For example, the error below is returned if the user provided a malformed bitcoin address when creating a new order:

```
{
    "error": "XMRTO-ERROR-002"
    "error_msg": "malformed btc address"
    "error_msg_display": "Is this a valid bitcoin address?"
}
```

List of all error codes

All possible XMR.to API error codes:

HTTP code	XMR.to error code	Error message/reason	Solution
503	XMRTO-ERROR-001	internal services not available	try again later
400	XMRTO-ERROR-002	malformed bitcoin address	check address validity
400	XMRTO-ERROR-003	invalid bitcoin amount	check amount data type
400	XMRTO-ERROR-004	bitcoin amount out of bounds	check min and max amount
400	XMRTO-ERROR-005	unexpected validation error	contact support
404	XMRTO-ERROR-006	requested order not found	check order UUID
503	XMRTO-ERROR-007	third party service not available	try again later
503	XMRTO-ERROR-008	insufficient funds available	try again later
400	XMRTO-ERROR-009	invalid request	check request parameters
400	XMRTO-ERROR-010	payment protocol failed	invalid or outdated data served by URL
400	XMRTO-ERROR-011	malformed payment protocol url	URL is malformed or cannot be contacted
403	XMRTO-ERROR-012	too many requests	try less often
403	XMRTO-ERROR-013	access forbidden	none
403	XMRTO-ERROR-014	service is not available in your region	none
400	XMRTO-ERROR-015	invalid monero amount	check amount data type
400	XMRTO-ERROR-016	invalid currency	check available currency options
400	XMRTO-ERROR-017	malformed lightning network invoice	provide a correct invoice for the main network
503	XMRTO-ERROR-018	lightning payment unlikely to succeed	check first if xmr.to has routes available
400	XMRTO-ERROR-019	lightning invoice preimage already known	don't use the same invoice more than once
400	XMRTO-ERROR-020	lightning invoice is expired or about to expire	create a new lightning invoice

1.1.4 Rate limiting

API endpoints might be rate-limited to protect them against excessive querying. If you are experiencing this, please adjust your querying behavior.

See the *List of all error codes* section for the relevant error code.

1.1.5 Region blocking

Depending on the regulatory landscape, XMR.to blocks certain regions in order to be compliant with local laws.

Forwarding User Information: If you acts as a proxy and send requests to our API of which you are not the originator, then you must forward the IP address of the user making the originating request, using the X-Forwarded-For HTTP header field to do so. See our API terms here: https://xmr.to/api-terms

See the *List of all error codes* section for the relevant error code.

1.1.6 Various parameters

Currently, XMR.to has a few additional parameters that are constant. Therefore, we do not expose them via a dedicated API endpoint. However, for the sake of completeness, we list them here:

Description	Value
Number of seconds after which an order times out if no or only partial payment occurred	900
Number of recommended mixins to use in Monero payments	11

1.2 Version 3

Version 3 is the successor of version 2 of the XMR.to API, active from April 2020.

Note: API version 3 is the current version.

Version 2 of the XMR.to API was deprecated in April 2020.

Version 2 of the XMR.to API was removed in August 2020.

Please see API version 3 migration guide when switching from API veirsion 2.

Compared to version 2, this API:

- Adds support for lightning network orders.
- Has generalized API field names.
- Changes the interface for order creation. Separating amount and currency in 2 different fields.
- Restructures the response for finished orders when tracking an order.
- Drops support for the long payment id (because of the 10/2019 protocol update).
- Drops support for XMR payment via integrated addresses.
- Drops support for deprecated field *xmr_required_amount* in favour of *incoming_amount_total*.
- Drops support for deprecated field *xmr_receiving_integrated_address* in favor of *receiving_subaddress*.
- Drops support for deprecated field *xmr_required_payment_id_long* in favor of *receiving_subaddress*.
- Drops support for deprecated field *xmr_required_payment_id_short* in favor of *receiving_subaddress*.
- Drops support for deprecated field *xmr_receiving_address* in favor of *receiving_subaddress*.
- Drops support for deprecated field *btc_num_confirmations_before_purge* in favor of *btc_num_confirmations_threshold*.

1.2.1 Changelog

Date	Change
Jun 2020	Added 2 new states to Orders (`REJECTED` and `PAYMENT_FAILED`).
Apr 2020	Activated this API version.
Mar 2020	Added two lightning network endpoints: `order_create_ln` and `order_ln_check_route`.
Nov 2019	Added endpoint to partially pay an underpaid order. Added `btc_amount_partial` field to response of `order_status_query`.
Oct 2019	`order_create` now accepts `currency` field.
Sep 2019	Created this API version.

1.2.2 Querying order parameters

API endpoint: https://xmr.to/api/v3/xmr2btc/order_parameter_query/

The order parameter endpoint supplies information about whether new orders can be created. In this case, this endpoint provides the current price, order limits, etc. for newly created orders.

Note: It is possible to query the status of existing orders even if the order parameter endpoint reports *not available*.

Request

Issue a *GET* request to query current order parameters.

Response

On success (*HTTP* code `200`), the request returns the following *JSON* data:

```
{
    "lower_limit": <lower_order_limit_in_btc_as_string>,
    "ln_lower_limit": <lightning_lower_order_limit_in_btc_as_string>,
    "ln_upper_limit": <lightning_upper_order_limit_in_btc_as_string>,
    "price": <price_of_1_xmr_in_btc_as_offered_by_service_as_string>,
    "upper_limit": <upper_order_limit_in_btc_as_string>,
    "zero_conf_enabled": <true_if_zero_conf_is_enabled_as_boolean>,
    "zero_conf_max_amount": <up_to_this_amount_zero_conf_is_possible_as_string>
}
```

Fields should be self-explanatory.

Errors

On failure, one of the following errors is returned:

- `XMRTO-ERROR-001`
- `XMRTO-ERROR-005`
- `XMRTO-ERROR-007`
- `XMRTO-ERROR-008`
- `XMRTO-ERROR-012`
- `XMRTO-ERROR-014`

For error details see the *List of all error codes* section.

Rate limitation

It is possible to request the endpoint up to **3 times per second**.

Example

Request

```
curl https://xmr.to/api/v3/xmr2btc/order_parameter_query/
```

or

```
http https://xmr.to/api/v3/xmr2btc/order_parameter_query/
```

Response

```
{
    "price": "0.017666",
    "upper_limit": "20",
    "lower_limit": "0.002",
    "ln_upper_limit": "0.0077",
    "ln_lower_limit": "0.00015",
    "zero_conf_enabled": true,
    "zero_conf_max_amount": "0.1"
}
```

1.2.3 Creating a new order

API endpoint: https://xmr.to/api/v3/xmr2btc/order_create/

The order creation endpoint allows to create a new order at the current price. The user has to supply a bitcoin destination address, amount and currency to create the order.

Note: Please use the `order_check_price` API endpoint if you only want to check the price for a specific Bitcoin/Monero amount.

Request

Issue a *POST* request to create a new order supplying the following parameters:

```
{
    "amount": <requested_amount_as_float>,
    "amount_currency": <currency_of_the_requested_amount_as_string>,
    "btc_dest_address": <requested_destination_address_as_string>
}
```

The available currencies for the `amount_currency` field are `BTC` and `XMR`.

Note: Make sure that `amount` is inside the possible limits for an order. These limits can be queried using the `order_parameter_query` endpoint.

Response

On success (*HTTP* code `201`, "created"), the request returns the following *JSON* data:

```
{
    "state": "TO_BE_CREATED",
    "btc_amount": <requested_amount_in_btc_as_string>,
    "btc_dest_address": <requested_destination_address_as_string>,
    "uses_lightning": <boolean_indicating_the_method_for_the_payment>,
    "uuid": <unique_order_identifier_as_12_character_string>
}
```

The field `state` reflects the state of an order. If `state` is `TO_BE_CREATED` in the response, the order has been registered for creation and you can use the order `uuid` to start querying the order's status. All other fields should be self-explanatory.

Note: Even though the order can be created by specifying an amount in XMR, the response will always return the amount in BTC based on the availabe rate at the time of the request.

Errors

On failure, one of the following errors is returned:

- `XMRTO-ERROR-001`
- `XMRTO-ERROR-002`
- `XMRTO-ERROR-003`
- `XMRTO-ERROR-004`
- `XMRTO-ERROR-005`
- `XMRTO-ERROR-012`
- `XMRTO-ERROR-014`
- `XMRTO-ERROR-015`
- `XMRTO-ERROR-016`

For error details see the *List of all error codes* section.

Rate limitation

It is possible to request the endpoint up to **20 times per minute**.

Example

In this example, we create an order for donating 0.1 BTC to the Monero developers (using Bitcoin, ironically).

Request

```
curl --data '{"btc_dest_address": "1FhnVJi2V1k4MqXm2nHoEbY5LV7FPai7bb", \
    "amount": 0.1, "amount_currency": "BTC"}' -H "Content-Type: application/json" \
    https://xmr.to/api/v3/xmr2btc/order_create/
```

or

```
http --json https://xmr.to/api/v3/xmr2btc/order_create/ btc_dest_
↪address=1FhnVJi2V1k4MqXm2nHoEbY5LV7FPai7bb amount=0.1 amount_currency=BTC
```

Hint: Remember to set the *HTTP* Content-Type to `application/json`!

Response

```
{
    "state": "TO_BE_CREATED",
    "btc_amount": "0.1",
    "btc_dest_address": "1FhnVJi2V1k4MqXm2nHoEbY5LV7FPai7bb",
    "uses_lightning": false,
    "uuid": "xmrto-XCZEsu"
}
```

1.2.4 Creating a new order using a payment protocol URL

API endpoint: https://xmr.to/api/v3/xmr2btc/order_create_pp/

This alternative order creation endpoint allows to create a new order at the current price, but instead of providing an explicit address and amount, the user provides a BIP70 url that once fetched by XMR.to will provide the address and amount.

Request

Issue a *POST* request to create a new order supplying the following parameters:

```
{
    "pp_url": <payment_protocol_url>
}
```

Note: XMR.to is able to correct automatically URLs provided by users to the correct one serving a BIP70-protocol answer. For instance, values such as `https://bitpay.com/invoice?id=xxx`, `bitcoin:?r=https://bitpay.com/i/xxx` will be corrected to the correct one automatically (the correct one being for *Bitpay*: `https://bitpay.com/i/KbMdd4EhnLXSbpWGKsaeo6`.

Response

On success (*HTTP* code `201`, "created"), the request returns the following *JSON* data:

```
{
    "state": "TO_BE_CREATED",
    "btc_amount": <requested_amount_in_btc_as_string>,
    "btc_dest_address": <requested_destination_address_as_string>,
    "uses_lightning": <boolean_indicating_the_method_for_the_payment>,
    "uuid": <unique_order_identifier_as_12_character_string>,
    "pp_url": <payment_bip70_protocol_url>
}
```

The field `state` reflects the state of an order. If `state` is `TO_BE_CREATED` in the response, the order has been registered for creation and you can use the order `uuid` to start querying the order's status. All other fields should be self-explanatory.

Errors

On failure, one of the following errors is returned:

- `XMRTO-ERROR-001`
- `XMRTO-ERROR-002`
- `XMRTO-ERROR-003`
- `XMRTO-ERROR-004`
- `XMRTO-ERROR-010`
- `XMRTO-ERROR-011`
- `XMRTO-ERROR-012`
- `XMRTO-ERROR-014`

For error details see the *List of all error codes* section.

Rate limitation

It is possible to request the endpoint up to **20 times per minute**.

Example

In this example, we create an order for donating 0.1 BTC to the Monero developers (using Bitcoin, ironically).

Request

```
curl --data '{"pp_url": "https://bitpay.com/invoice?id=<invoice_id>"}' -H "Content-
↪Type: application/json" https://xmr.to/api/v3/xmr2btc/order_create_pp/
```

or

```
http --json https://xmr.to/api/v3/xmr2btc/order_create_pp/ pp_url="https://bitpay.com/
↪invoice?id=<invoice_id>"
```

Hint: Remember to set the *HTTP* Content-Type to `application/json`!

Response

```
{
    "state": "TO_BE_CREATED",
    "btc_amount": "0.1",
    "btc_dest_address": "1FhnVJi2V1k4MqXm2nHoEbY5LV7FPai7bb",
    "uses_lightning": false,
    "uuid": "xmrto-XCZEsu",
    "pp_url": "https://bitpay.com/i/xxx"
}
```

1.2.5 Creating a new order using a lightning network invoice

API endpoint: https://xmr.to/api/v3/xmr2btc/order_create_ln/

This alternative order creation endpoint allows to create a new order at the current price, where the amount will be sent through the lightning network instead of a normal onchain transaction. The user provides a BOLT11 payment request that once decoded by XMR.to will provide all the details required for the payment (destination public key, amount, etc).

Request

Issue a *POST* request to create a new order supplying the following parameters:

```
{
    "ln_invoice": <lighting_network_incoice>
}
```

Note: You should first check if XMR.to has routes available to pay the provided invoice, using `order_ln_check_route`. If no routes exist or the probability of success is very low, the request will fail.

Response

On success (*HTTP* code `201`, "created"), the request returns the following *JSON* data:

```
{
    "state": "TO_BE_CREATED",
    "btc_amount": <requested_amount_in_btc_as_string>,
    "btc_dest_address": <requested_destination_publickey_as_string>,
    "uses_lightning": <boolean_indicating_the_method_for_the_payment>,
    "msat_amount": <request_amount_in_msat_as_integer>,
    "uuid": <unique_order_identifier_as_12_character_string>
}
```

The field `state` reflects the state of an order. If `state` is `TO_BE_CREATED` in the response, the order has been registered for creation and you can use the order `uuid` to start querying the order's status. All other fields should be self-explanatory.

Errors

On failure, one of the following errors is returned:

- `XMRTO-ERROR-001`
- `XMRTO-ERROR-004`
- `XMRTO-ERROR-012`
- `XMRTO-ERROR-014`
- `XMRTO-ERROR-017`
- `XMRTO-ERROR-018`
- `XMRTO-ERROR-019`
- `XMRTO-ERROR-020`

For error details see the *List of all error codes* section.

Rate limitation

It is possible to request the endpoint up to **20 times per minute**.

Example

In this example, we create an order using an example invoice.

Request

```
curl --data '{"ln_invoice":
↪"lnbc2500u1pvjluezpp5qqqsyqcyq5rqwzqfqqqsyqcyq5rqwzqfqqqsyqcyq5rqwzqfqypqdpquwpc4curk03c9wlrswe78q4
↪"}' -H "Content-Type: application/json" https://xmr.to/api/v3/xmr2btc/order_create_
↪ln/
```

or

```
http --json https://xmr.to/api/v3/xmr2btc/order_create_ln/ ln_invoice=
↪"lnbc2500u1pvjluezpp5qqqsyqcyq5rqwzqfqqqsyqcyq5rqwzqfqqqsyqcyq5rqwzqfqypqdpquwpc4curk03c9wlrswe78q4
↪"
```

(continues on next page)

(continued from previous page)

Hint: Remember to set the *HTTP* Content-Type to `application/json`!

Response

```
{
    "state": "TO_BE_CREATED",
    "btc_amount": "0.0025",
    "btc_dest_address":
↪"03e7156ae33b0a208d0744199163177e909e80176e55d97a2f221ede0f934dd9ad",
    "uses_lightning": true,
    "msat_amount": 250000000,
    "uuid": "xmrto-XCZEsu",
    "pp_url": "https://bitpay.com/i/xxx"
}
```

1.2.6 Querying order status

API endpoint: https://xmr.to/api/v3/xmr2btc/order_status_query/

The order status endpoint allows users to query the status of an order, thereby obtaining payment details and order processing progress.

Request

Issue a *POST* request to query the status of a given order. You have to supply the order's `uuid` in the request:

```
{
    "uuid": <unique_order_identifier_as_12_character_string>,
}
```

Response

On success (*HTTP* code `200`), the request returns the following *JSON* data:

```
{
    "state": <order_state_as_string>,
    "btc_amount": <requested_amount_in_btc_as_string>,
    "btc_amount_partial": <partial_amount_in_btc_as_string>,
    "btc_dest_address": <requested_destination_address_as_string>,
    "uuid": <unique_order_identifier_as_12_character_string>
    "btc_num_confirmations_threshold": <btc_num_confirmations_threshold_as_integer>,
    "created_at": <timestamp_as_string>,
    "expires_at": <timestamp_as_string>,
    "seconds_till_timeout": <seconds_till_timeout_as_integer>,
    "incoming_amount_total": <amount_in_incoming_currency_for_this_order_as_string>,
    "remaining_amount_incoming": <amount_in_incoming_currency_that_the_user_must_
↪still_send_as_string>,
    "incoming_num_confirmations_remaining": <num_incoming_currency_confirmations_
↪remaining_before_bitcoins_will_be_sent_as_integer>,
```

(continues on next page)

(continued from previous page)

```
    "incoming_price_btc": <price_of_1_incoming_in_btc_currency_as_offered_by_service_
↪as_string>,
    "receiving_subaddress": <xmr_subaddress_user_needs_to_send_funds_to_as_string>,
    "recommended_mixin": <recommended_mixin_as_integer>,
    "uses_lightning": <boolean_indicating_the_method_for_the_payment>,
    "msat_amount": <order_amount_in_msat_as_integer>,
    "payments": [<payment_objects>]
}
```

Note: The field *btc_amount_partial* is added to allow partial payments of an underpaid order.

Note: Since lightning network invoices and payments make use of millisatoshi (1sat/1000), the field `msat_amount` was added to make it clear the exact amount being transferred.

The user has to pay the order using the Monero subaddress.

Presence of some of these fields depend on `state`, which can take the following values:

Value	Meaning
TO_BE_CREATED	order creation pending
REJECTED	order will not be processed, extra validation failed
FLAGGED_DESTINATION_ADDRESS	order will not be processed, address does not meet requirements
UNPAID	waiting for Monero payment by user
UNDERPAID	order partially paid
PAID_UNCONFIRMED	order paid, waiting for enough confirmations
PAID	order paid and sufficiently confirmed
BTC_SENT	bitcoin payment sent
TIMED_OUT	order timed out before payment was complete
PAYMENT_FAILED	attempt to send the payment failed, ask for a refund

The `payments` should be empty until the order reaches the status `BTC_SENT`, at that moment an item with the payment information is added.

For standard on chain orders the structure is the following:

```
{
    "method": <order_method_as_string>,
    "amount": <sent_amount_in_btc_as_string>,
    "num_confirmations": <btc_num_confirmations_as_integer>,
    "tx_id": <btc_transaction_id_as_string>
}
```

Lightning network payments contain `payment_hash` and `payment_preimage` fields instead of `num_confirmations` and `tx_id`:

```
{
    "method": <order_method_as_string>,
    "amount": <sent_amount_in_btc_as_string>,
    "amount_msat": <sent_amount_in_msat_as_integer>,
    "payment_hash": <payment_hash_included_in_the_invoice_as_string>,
    "payment_preimage": <preimage_that_generates_the_hash_as_string>
}
```

In case the order cannot be found, the appropriate error is returned (see below).

All other fields should be self-explanatory.

Errors

On failure, one of the following errors is returned:

- `XMRTO-ERROR-006`
- `XMRTO-ERROR-009`
- `XMRTO-ERROR-012`
- `XMRTO-ERROR-014`

For error details see the *List of all error codes* section.

Rate limitation

It is possible to request the endpoint up to **3 times per second**.

Example

Continuing from our previous example, we can query the order by supplying the order's unique identifier `uuid`.

Request

```
curl --data '{"uuid": "xmrto-VkT2yM"}' -H "Content-Type: application/json" \
    https://xmr.to/api/v3/xmr2btc/order_status_query/
```

or

```
http --json https://xmr.to/api/v3/xmr2btc/order_status_query/ uuid=xmrto-VkT2yM
```

Response

The response gives the current status of the order:

```
{
    "incoming_price_btc": "0.01760396",
    "uuid": "xmrto-XCZEsu",
    "state_str": "BTC_SENT",
    "btc_amount": "0.1",
    "btc_amount_partial": "0",
    "btc_dest_address": "1FhnVJi2V1k4MqXm2nHoEbY5LV7FPai7bb",
    "receiving_subaddress":
↪"83BGzCTthheE2KxNTBPnPJjJUthYPfDfCf3ENSVQcpga8RYSxNz9qCz1qp9MLye9euMjckGi11cRdeVGqsVqTLgH8w5fJ1D
↪",
    "created_at": "2017-07-01T08:11:27Z"
    "expires_at": "2017-07-01T08:26:27Z",
    "seconds_till_timeout": 857,
    "incoming_amount_total": "5.68",
    "remaining_amount_incoming": "5.68",
    "incoming_num_confirmations_remaining": -1,
    "recommended_mixin": 4,
    "btc_num_confirmations_threshold": 144,
```

(continues on next page)

(continued from previous page)

```
    "payments": [
        {
            "method": "On Chain",
            "amount": "0.001",
            "num_confirmations": 0,
            "tx_id": "320e501c29778ce280f69ba3d1564f0e890f478752cd3b59d36e2801ccb972f2
↪"
        }
    ],
    "uses_lightning": False
}
```

In this example, the next step would require the user to pay *5.68* XMR to the Monero subaddress *83BGzC...*.

Note: The payment **must** be made before the order expires, in this case, inside *857* seconds.

Note: The field *btc_amount_partial* is added to allow partial payments of an underpaid order.

1.2.7 Order partial payment

API endpoint: https://xmr.to/api/v3/xmr2btc/order_partial_payment/

The order partial payment endpoint allows users to partially pay an underpaid order, thereby confirming that less BTC will arrive at the destination address.

Request

Issue a *POST* request to partially pay an underpaid order. You have to supply the order's `uuid` in the request:

```
{
    "uuid": <unique_order_identifier_as_12_character_string>,
}
```

Response

On success (*HTTP* code `200`), the request does not return any more data.

The order needs to be underpaid (*state* is `UNDERPAID`), in oder to confirm a partial payment for this very order.

To get information on the updated order please trigger `order_status_query` again.

The user has to pay the order using the integrated address or the Monero subaddress. In case the user's wallet does not support integrated addresses, the user can pay via the old-style address while specifying the long payment id.

In case the order cannot be found, the appropriate error is returned (see below).

The value of `btc_amount_partial` will be *0* for every order state but `UNDERPAID`. In case the orderis `UNDERPAID`, `btc_amount_partial` will be populated with the appropriate BTC amount according to the XMR amount paid and your personal rate.

Errors

On failure, one of the following errors is returned:

- `XMRTO-ERROR-006`
- `XMRTO-ERROR-009`
- `XMRTO-ERROR-012`
- `XMRTO-ERROR-014`

For error details see the *List of all error codes* section.

Rate limitation

It is possible to request the endpoint up to **3 times per second**.

Example

Continuing from our previous example, imagine the order `xmrto-VkT2yM` was underpaid with *5* XMR. Remember you had to pay *5.68* XMR?

In this case `remaining_amount_incoming` will change to *0.68* XMR while `btc_amount_partial` will return approximately *0.088* BTC - Your personal rate stays the same, `incoming_price_btc=0.01760396`. You will see the updated order information with the next post to `order_status_query`.

You now have the option to partially pay your order by supplying the order's unique identifier `uuid` to the endpoint.

Request

```
curl --data '{"uuid": "xmrto-VkT2yM"}' -H "Content-Type: application/json" \
    https://xmr.to/api/v3/xmr2btc/order_partial_payment/
```

or

```
http --json https://xmr.to/api/v3/xmr2btc/order_partial_payment/ uuid=xmrto-VkT2yM
```

Response

The response gives no detailed information. However, the next call to `order_status_query` returns the current/updated status of the order.

Note: A partial payment **must** be made before the order expires.

Note: The field *btc_amount_partial* is added to allow partial payments of an underpaid order.

1.2.8 Querying order price

API endpoint: https://xmr.to/api/v3/xmr2btc/order_check_price/

The order status endpoint allows users to query the recent price for a given amount in BTC/XMR.

Request

Issue a *POST* request to query the price of a BTC/XMR amount. The user has to the amount and currency to create the order.

```
{
    "amount": <requested_amount_as_float>,
    "amount_currency": <currency_of_the_requested_amount>,
}
```

The available currencies for the `amount_currency` field are `BTC` and `XMR`.

Note: Make sure that `amount` is inside the possible limits for an order. These limits can be queried using the `order_parameter_query` endpoint.

Response

On success (*HTTP* code `200`), the request returns the following *JSON* data:

```
{
    "btc_amount": <requested_amount_in_btc_as_string>,
    "incoming_amount_total": <amount_in_incoming_currency_for_this_order_as_string>,
    "incoming_num_confirmations_remaining": <num_incoming_currency_confirmations_
↪remaining_before_bitcoins_will_be_sent_as_integer>,
    "incoming_price_btc": <price_of_1_incoming_in_btc_as_offered_by_service_as_string>
}
```

Errors

On failure, one of the following errors is returned:

- `XMRTO-ERROR-001`
- `XMRTO-ERROR-003`
- `XMRTO-ERROR-004`
- `XMRTO-ERROR-005`
- `XMRTO-ERROR-007`
- `XMRTO-ERROR-012`
- `XMRTO-ERROR-014`
- `XMRTO-ERROR-015`
- `XMRTO-ERROR-016`

For error details see the *List of all error codes* section.

Rate limitation

It is possible to request the endpoint **once every 3 seconds**.

Example

Imagine we want to check the recent price for an order including the payment of *0.15* BTC.

Request

```
curl --data '{"amount": 0.15, "amount_currency": "BTC"}' \
    -H "Content-Type: application/json" \
    https://xmr.to/api/v3/xmr2btc/order_check_price/
```

or

```
http --json https://xmr.to/api/v3/xmr2btc/order_check_price/ btc_amount=0.15
```

Imagine we want to check the recent price for an order including the payment of *10* XMR.

```
curl --data '{"amount": 10, "amount_currency": "XMR"}' \
    -H "Content-Type: application/json" \
    https://xmr.to/api/v3/xmr2btc/order_check_price/
```

or

```
http --json https://xmr.to/api/v3/xmr2btc/order_check_price/ xmr_amount=10
```

Response

The response gives the current price for the order (when paying *0.15* BTC):

```
{
    "btc_amount": "0.15",
    "incoming_amount_total": "21.4185",
    "incoming_num_confirmations_remaining": 1,
    "incoming_price_btc": "0.00700329"
}
```

In the above example, the order including the payment of *0.15* BTC would require the user to pay *21.4185* XMR.

The response gives the current price for the order (when paying *10* XMR):

```
{
    "btc_amount": "0.0701264",
    "incoming_amount_total": "10",
    "incoming_num_confirmations_remaining": 0,
    "incoming_price_btc": "0.00701264"
}
```

In the above example, the order including the payment of *10* XMR would result in an order that eventually pays *0.0701264* BTC to the destination address.

Note: Even though the price can be queried by specifying an amount in XMR, the response will always return the amount in BTC based on the availabe rate at the time of the request.

1.2.9 Querying lightning routes

API endpoint: https://xmr.to/api/v3/xmr2btc/order_ln_check_route/

The route checking endpoint allows users to know if XMR.to is able to pay their lighting invoice, before trying to create a new order.

Request

Issue a *GET* request with a `ln_invoice` query_param containing your invoice.

Response

On success (*HTTP* code `200`), the request returns the following *JSON* data:

```
{
    "success_probability": <success_prob_between_0_and_1>,
    "num_routes": <number_of_existing_routes_to_destination_node>
}
```

Errors

On failure, one of the following errors is returned:

- `XMRTO-ERROR-001`
- `XMRTO-ERROR-004`
- `XMRTO-ERROR-012`
- `XMRTO-ERROR-014`
- `XMRTO-ERROR-017`

For error details see the *List of all error codes* section.

Rate limitation

It is possible to request the endpoint **once every 3 seconds**.

Example

Request

```
curl https://xmr.to/api/v3/xmr2btc/order_parameter_query/?ln_
↪invoice=lnbc2500u1pvjluezpp5qqqsyqcyq5rqwzqfqqqsyqcyq5rqwzqfqqqsyqcyq5rqwzqfqypqdpquwpc4curk03c9wlr
```

or

```
http https://xmr.to/api/v3/xmr2btc/order_parameter_query/?ln_
↪invoice=lnbc2500u1pvjluezpp5qqqsyqcyq5rqwzqfqqqsyqcyq5rqwzqfqqqsyqcyq5rqwzqfqypqdpquwpc4curk03c9wlr
```

Response

```
{
    "success_probability": 0.95,
    "num_routes": 1
}
```

1.3 API migration

This is about migrating from API version 2 to API version 3 of the XMR.to API.

This document describes changes in already existing API calls.

Please see API version 3 for more details and new features.

Note: API version 3 is the current version.

1.3.1 Querying order parameters

Request

Issue a *GET* request to query current order parameters.

Response

Both versions of the API return:

```
{
    "lower_limit": <lower_order_limit_in_btc>,
    "price": <price_of_1_xmr_in_btc_as_offered_by_service>,
    "upper_limit": <upper_order_limit_in_btc>,
    "zero_conf_enabled": <true_if_zero_conf_is_enabled_as_boolean>,
    "zero_conf_max_amount": <up_to_this_amount_zero_conf_is_possible>
}
```

Note: API version 2 returns `lower_limit`, `upper_limit`, `price` and `zero_conf_max_amount` as `float`. API version 3 returns `lower_limit`, `upper_limit`, `price` and `zero_conf_max_amount` as `string`.

Additionally, with API version 3 you will receive:

```
{
    "ln_lower_limit": <lightning_lower_order_limit_in_btc_as_string>,
    "ln_upper_limit": <lightning_upper_order_limit_in_btc_as_string>
}
```

These are the operation limits for orders to be paid through the lighting network.

Example (API version 3)

Request

```
curl --url https://xmr.to/api/v3/xmr2btc/order_parameter_query/
```

Response

```
{
    "price": "0.017666",  // string
    "upper_limit": "20",  // string
    "lower_limit": "0.002",  // string
    "ln_upper_limit": "0.0077",
    "ln_lower_limit": "0.00015",
    "zero_conf_enabled": true,
    "zero_conf_max_amount": "0.1"  // string
}
```

1.3.2 Creating a new order

Request

Issue a *POST* request to create a new order supplying the following parameters.

With API version 2 there are separate parameters when creating orders by specifying the order amount in *BTC* or *XMR*, respectively:

```
{
    "btc_amount": <requested_amount_in_btc_as_float>,
    "btc_dest_address": <requested_destination_address_as_string>
}
```

or

```
{
    "xmr_amount": <requested_amount_in_xmr_as_float>,
    "btc_dest_address": <requested_destination_address_as_string>
}
```

With API version 3 the parameters look like this:

```
{
    "amount": <requested_amount_as_float>,
    "amount_currency": <currency_of_the_requested_amount_as_string>,
    "btc_dest_address": <requested_destination_address_as_string>
}
```

The available currencies for the `amount_currency` field are `BTC` and `XMR`.

Response

Both versions of the API return:

```
{
    "state": "TO_BE_CREATED",
    "btc_amount": <requested_amount_in_btc>,
    "btc_dest_address": <requested_destination_address_as_string>,
    "uuid": <unique_order_identifier_as_12_character_string>
}
```

Note: API version 2 returns `btc_amount` as `float`. API version 3 returns `btc_amount` as `string`.

Additionally with API version 3 you will receive:

```
{
    "uses_lightning": <boolean_indicating_the_method_for_the_payment>
}
```

This indicates the payment method, either onchain or offchain via lightning.

Example (API version 3)

Request

```
curl --data '{"btc_dest_address": "1FhnVJi2V1k4MqXm2nHoEbY5LV7FPai7bb", \
    "amount": 0.1, "amount_currency": "BTC"}' -H "Content-Type: application/json" \
    --url https://xmr.to/api/v3/xmr2btc/order_create/
```

Response

```
{
    "state": "TO_BE_CREATED",
    "btc_amount": "0.1",  // string
    "btc_dest_address": "1FhnVJi2V1k4MqXm2nHoEbY5LV7FPai7bb",
    'uses_lightning': False,
    "uuid": "xmrto-XCZEsu"
}
```

1.3.3 Querying order status

Request

Issue a *POST* request to to query the status of a given order supplying the following parameters.

Both versions of the API require:

```
{
    "uuid": <unique_order_identifier_as_12_character_string>
}
```

Response

Please see API version 3 for more details.

Both versions of the API return:

```
{
    "state": <order_state_as_string>,
    "btc_amount": <requested_amount_in_btc>,
    "btc_amount_partial": <partial_amount_in_btc>,
    "btc_dest_address": <requested_destination_address_as_string>,
    "uuid": <unique_order_identifier_as_12_character_string>
    "btc_num_confirmations_threshold": <btc_num_confirmations_threshold_as_integer>,
    "created_at": <timestamp_as_string>,
    "expires_at": <timestamp_as_string>,
    "seconds_till_timeout": <seconds_till_timeout_as_integer>,
}
```

Note: API version 2 returns `btc_amount` and `btc_amount_partial` as `float`. API version 3 returns `btc_amount` and `btc_amount_partial` as `string`.

With API version 3 some keys have been removed (Mostly due to the fact that we do not support integrated addresses anymore.

- *xmr_receiving_address*
- *xmr_receiving_integrated_address*
- *xmr_required_payment_id_long*
- *xmr_required_payment_id_short*
- *xmr_required_amount* (deprecated)
- *btc_num_confirmations_before_purge* in favour of *btc_num_confirmations_threshold* only.

With API version 3 some keys have been renamed to more general names:

API version 2	API version 3
`xmr_amount_total` <*float*>	`incoming_amount_total` <*string*>
`xmr_amount_remaining` <*float*>	`remaining_amount_incoming` <*string*>
`xmr_num_confirmations_remaining`	`incoming_num_confirmations_remaining`
`xmr_price_btc` <*float*>	`incoming_price_btc` <*string*>
`xmr_receiving_subaddress`	`receiving_subaddress`
`xmr_recommended_mixin`	`recommended_mixin`

Additionally, with API version 3 you will receive:

```
{
    "uses_lightning": <boolean_indicating_the_method_for_the_payment>,
    "payments": [<payment_objects>]
}
```

`uses_lightning` indicates the payment method, either onchain or offchain via lightning.

The `payment_object` is new with API version 3. Some keys have been restructured (For *On Chain* payments.):

API version 2	API version 3
`btc_transaction_id`	`payments` - empty (`[]`) if order not paid - otherwise `<payment_objects>`
`btc_num_confirmations`	

The available payment methods for the `method` field of a `payment_object` are *On Chain* and *Lightning Network*.

Please see API version 3 for more details.

Example (API version 3)

Request

```
curl --data '{"uuid": "xmrto-VkT2yM"}' -H "Content-Type: application/json" \
    --url https://xmr.to/api/v3/xmr2btc/order_status_query/
```

Response

```
{
    "incoming_price_btc": "0.01760396",
    "uuid": "xmrto-XCZEsu",
    "state_str": "BTC_SENT",
    "btc_amount": "0.1",  // string
    "btc_amount_partial": "0",  // string
    "btc_dest_address": "1FhnVJi2V1k4MqXm2nHoEbY5LV7FPai7bb",
    "receiving_subaddress":
↪"83BGzCTthheE2KxNTBPnPJjJUthYPfDfCf3ENSVQcpga8RYSxNz9qCz1qp9MLye9euMjckGi11cRdeVGqsVqTLgH8w5fJ1D
↪",
    "created_at": "2017-07-01T08:11:27Z"
    "expires_at": "2017-07-01T08:26:27Z",
    "seconds_till_timeout": 857,
    "incoming_amount_total": "5.68",
    "remaining_amount_incoming": "5.68",
    "incoming_num_confirmations_remaining": -1,
    "recommended_mixin": 4,
    "btc_num_confirmations_threshold": 144,
    "payments": [
        {
            "method": "On Chain",
            "amount": "0.001",
            "num_confirmations": 0,
            "tx_id": "320e501c29778ce280f69ba3d1564f0e890f478752cd3b59d36e2801ccb972f2
↪"
        }
    ],
    "uses_lightning": False
}
```

1.3.4 Querying order price

Request

Issue a *POST* request to query the price of a BTC/XMR amount.

With API version 2 there are separate parameters when creating orders by specifying the order amount in *BTC* or *XMR*, respectively:

```
{
    "btc_amount": <requested_amount_in_btc_as_float>
}
```

or

```
{
    "xmr_amount": <requested_amount_in_xmr_as_float>
}
```

With API version 3 the parameters look like this:

```
{
    "amount": <requested_amount_as_float>,
    "amount_currency": <currency_of_the_requested_amount_as_string>
}
```

The available currencies for the `amount_currency` field are `BTC` and `XMR`.

Response

Both versions of the API return:

```
{
    "btc_amount": <requested_amount_in_btc>,
}
```

Note: API version 2 returns `btc_amount` as `float`. API version 3 returns `btc_amount` as `string`.

With API version 3 some keys have been renamed to more general names:

API version 2	API version 3
`xmr_amount_total` <*float*>	`incoming_amount_total` <*string*>
`xmr_num_confirmations_remaining`	`incoming_num_confirmations_remaining`
`xmr_price_btc` <*float*>	`incoming_price_btc` <*string*>

Example (API version 3)

Request

Imagine we want to check the recent price for an order including the payment of *0.15* BTC.

Request

```
curl --data '{"amount": 0.15, "amount_currency": "BTC"}' \
    -H "Content-Type: application/json" \
    https://xmr.to/api/v3/xmr2btc/order_check_price/
```

Imagine we want to check the recent price for an order including the payment of *10* XMR.

```
curl --data '{"amount": 10, "amount_currency": "XMR"}' \
    -H "Content-Type: application/json" \
    https://xmr.to/api/v3/xmr2btc/order_check_price/
```

Response

The response gives the current price for the order (when paying *0.15* BTC):

```
{
    "btc_amount": "0.15",  // string
    "incoming_amount_total": "21.4185",  // string
    "incoming_num_confirmations_remaining": 1,
    "incoming_price_btc": "0.00700329"  // string
}
```

In the above example, the order including the payment of *0.15* BTC would require the user to pay *21.4185* XMR.

The response gives the current price for the order (when paying *10* XMR):

```
{
    "btc_amount": "0.0701264",  // string
    "incoming_amount_total": "10",  // string
    "incoming_num_confirmations_remaining": 0,
    "incoming_price_btc": "0.00701264"  // string
}
```

In the above example, the order including the payment of *10* XMR would result in an order that eventually pays *0.0701264* BTC to the destination address.

1.4 Public test instance

XMR.to offers a public test instance to ease development on services depending on its API. The instance is connected to the Monero stage network and Bitcoin test network. (The Monero test network is intended for protocol development and is not suitable to test service integration).

You can create an order, send Monero stagenet coins to XMR.to, and will receive Bitcoin testnet coins on successful order execution.

Hint: The addresses used look different from mainnet ones: Monero stagenet addresses start with *5*, while Bitcoin testnet addresses start with *m*, *n* or *2*.

1.4.1 Access

The public test instance is accessible under https://test.xmr.to

We do not have a Tor onion address at the moment. You can still use https://test.xmr.to/nojs/ in the Tor browser, however.

1.4.2 Funding

As there is no BTC/XMR exchange available on the stagenet, the hotwallet of the public test instance cannot be automatically refilled. If the service runs low (i.e., the max order is very small), simply send Bitcoin testnet coins to the following address:

```
2N5AYGnYKM7zgTe1n8P7mjUE3DavD1ub7Zs
```

1.5 Problems?

Please check:

- Are you including the proper parameters?
- Are you using the proper request type *POST* vs. *GET*?
- Are you setting `"Content-Type: application/json"` in headers?
- Getting redirected? Add a `/` at the end of the API endpoint!
- Check your error responses - there should be a detailed error message.

If none of this resolves the problem, please contact support.

1.5.1 Contact

- Follow us on Twitter: https://twitter.com/xmr_to
- XMR.to support: support@xmr.to

CHAPTER

TWO

PREFACE

Author XMR.to support

Contact support@xmr.to

Organization XMR.to

Abstract XMR.to - Pay the world with Monero. Monero is a secure, private, untraceable currency that is open-source and freely available to all. XMR.to converts moneroj (XMR) provided by the user to bitcoins (BTC), which are sent to a given bitcoin address.

XMR.to provides an API that enables developers to use the service in programs, apps, scripts etc. This API therefore allows developers to integrate an option to pay any bitcoin address in their product, for example, in a Monero wallet service. The API is a REST-like API using *JSON* as data format. It does not require authentication. This document describes this API and gives examples of its usage.