XMR.to API Documentation

Release 2.0

XMR.to

Sep 04, 2019

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 2.

CHAPTER 1

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 by accessing this API in any way or form. Read the Terms of Service here: https://xmr.to/terms-of-service

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 `v2`), followed by the conversion direction (currently only `xmr2btc`). Therefore, the complete API base URL is `https://xmr.to/api/v2/xmr2btc/`.

Note: The current version of the API is 2.

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 2 the complete URL would be `https://xmr.to/api/v2/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_error_message_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. 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"
}
```

List of all errors 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

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 errors 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.

See the *List of all errors 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 BTC confirmations required before we purge an order	144
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 2

Version 2 is the successor of version 1 of the XMR.to API. It is active since July 2017. Compared to version 1, this API supports:

- integrated addresses
- zero-conf transactions

Version 2 supports Monero subaddresses. .. note:

```
API version 2 is the current version.
```

1.2.1 Changelog

Date	Change
Sep 2019	Added xmr_receiving_subaddress field to response of `order_status_query`.
Mar 2019	Added error codes for region blocking.
Jan 2019	Added pp_url field to the response of Bitcoin payment endpoint.
Dec 2018	Added new endpoint `order_check_price`.
Dec 2018	Added rate limitation on API endpoints.
Aug 2018	Added Terms of Service requirement.
Feb 2018	Added endpoint for paying Bitcoin payment requests.

1.2.2 Querying order parameters

API endpoint: https://xmr.to/api/v2/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_float>,
    "price": <price_of_1_btc_in_xmr_as_offered_by_service_as_float>,
    "upper_limit": <upper_order_limit_in_btc_as_float>,
    "zero_conf_enabled": <true_if_zero_conf_is_enabled_as_boolean>,
    "zero_conf_max_amount": <up_to_this_amount_zero_conf_is_possible_as_float>
}
```

Fields should be self-explanatory.

Errors

On failure, one of the following errors is returned:

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

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

Rate limitation

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

Example

Request

```
curl https://xmr.to/api/v2/xmr2btc/order_parameter_query/
```

or

```
http https://xmr.to/api/v2/xmr2btc/order_parameter_query/
```

Response

```
{
    "price": 0.017666,
    "upper_limit": 20.0,
    "lower_limit": 0.002,
    "zero_conf_enabled": true,
    "zero_conf_max_amount": 0.1
}
```

1.2.3 Creating a new order

API endpoint: https://xmr.to/api/v2/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 and amount 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 amount.

Request

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

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

Note: Make sure that `btc_amount` 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_float>,
    "btc_dest_address": <requested_destination_address_as_string>,
    "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-002`
- `XMRTO-ERROR-003`
- `XMRTO-ERROR-004`
- `XMRTO-ERROR-005`
- `XMRTO-ERROR-012`
- `XMRTO-ERROR-014`

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

Rate limitation

It is possible to request the endpoint up to **4 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", \
    "btc_amount": 0.1}' -H "Content-Type: application/json" https://xmr.to/api/v2/
↪xmr2btc/order_create/
```

or

```
http --json https://xmr.to/api/v2/xmr2btc/order_create/ btc_dest_
↪address=1FhnVJi2V1k4MqXm2nHoEbY5LV7FPai7bb btc_amount=0.1
```

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

Response

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

1.2.4 Creating a new order using a payment protocol URL

API endpoint: https://xmr.to/api/v2/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_float>,
    "btc_dest_address": <requested_destination_address_as_string>,
    "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-005`
- `XMRTO-ERROR-010`
- `XMRTO-ERROR-011`
- `XMRTO-ERROR-012`
- `XMRTO-ERROR-014`

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

Rate limitation

It is possible to request the endpoint up to **4 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/v2/xmr2btc/order_create_pp/
```

or

```
http --json https://xmr.to/api/v2/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",
    "uuid": "xmrto-XCZEsu",
    "pp_url": "https://bitpay.com/i/xxx"
}
```

1.2.5 Querying order status

API endpoint: https://xmr.to/api/v2/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_float>,
    "btc_dest_address": <requested_destination_address_as_string>,
    "uuid": <unique_order_identifier_as_12_character_string>
    "btc_num_confirmations": <btc_num_confirmations_as_integer>,
    "btc_num_confirmations_before_purge": <btc_num_confirmations_before_purge_as_
↪integer>,
    "btc_transaction_id": <btc_transaction_id_as_string>,
    "created_at": <timestamp_as_string>,
    "expires_at": <timestamp_as_string>,
    "seconds_till_timeout": <seconds_till_timeout_as_integer>,
    "xmr_amount_total": <amount_in_xmr_for_this_order_as_float>,
    "xmr_amount_remaining": <amount_in_xmr_that_the_user_must_still_send_as_float>,
    "xmr_num_confirmations_remaining": <num_xmr_confirmations_remaining_before_
↪bitcoins_will_be_sent_as_integer>,
    "xmr_price_btc": <price_of_1_btc_in_xmr_as_offered_by_service_as_float>,
    "xmr_receiving_subaddress": <xmr_subaddress_user_needs_to_send_funds_to_as_string>
↪,
    "xmr_receiving_address": <xmr_old_style_address_user_can_send_funds_to_as_string>,
    "xmr_receiving_integrated_address": <xmr_integrated_address_user_needs_to_send_
↪funds_to_as_string>,
    "xmr_recommended_mixin": <xmr_recommended_mixin_as_integer>,
    "xmr_required_amount": <xmr_amount_user_needs_to_send_as_float>, (deprecated)
    "xmr_required_payment_id_long": <xmr_payment_id_user_needs_to_include_when_using_
↪old_stlye_address_as_string>
```

(continues on next page)

(continued from previous page)

```
    "xmr_required_payment_id_short": <xmr_payment_id_included_in_integrated_address_
↪as_string>
}
```

Note: The field *xmr_required_amount* is deprecated in favor of *xmr_amount_total*. The field *xmr_receiving_integrated_address* is deprecated in favor of *xmr_receiving_subaddress*. The field *xmr_required_payment_id_long* is deprecated in favor of *xmr_receiving_subaddress*. The field *xmr_required_payment_id_short* is deprecated in favor of *xmr_receiving_subaddress*. The field *xmr_receiving_address* is deprecated in favor of *xmr_receiving_subaddress*.

The user has to pay the order using the integrated address or the 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.

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

Value	Meaning
`TO_BE_CREATED`	order creation pending
`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
`NOT_FOUND`	order wasn't found in system (never existed or was purged)

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 errors 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/v2/xmr2btc/order_status_query/
```

or

```
http --json https://xmr.to/api/v2/xmr2btc/order_status_query/uuid=xmrto-VkT2yM
```

Response

The response gives the current status of the order:


```
{
    "xmr_price_btc": 0.01760396,
    "uuid": "xmrto-XCZEsu",
    "state_str": "UNPAID",
    "btc_amount": 0.1,
    "btc_dest_address": "1FhnVJi2V1k4MqXm2nHoEbY5LV7FPai7bb",
    "xmr_required_amount": 5.68054,
    "xmr_receiving_subaddress":
↪"83BGzCTthheE2KxNTBPnPJjJUthYPfDfCf3ENSVQcpga8RYSxNz9qCz1qp9MLye9euMjckGi11cRdeVGqsVqTLgH8w5fJ1D
↪",
    "xmr_receiving_address":
↪"44TVPcCSHebEQp4LnapPkhb2pondb2Ed7GJJLc6TkKwtSyumUnQ6QzkCCkojZycH2MRfLcujCM7QR1gdnRULRraV4UpB5n4
↪",
    "xmr_receiving_integrated_address":
↪"4EAAQR1vtv7EQp4LnapPkhb2pondb2Ed7GJJLc6TkKwtSyumUnQ6QzkCCkojZycH2MRfLcujCM7QR1gdnRULRraV64LYEHMdk
↪",
    "xmr_required_payment_id_long":
↪"356620a8410a4c683eda9b43fdc7fa531b721d70856c95994636361aafbda052",
    "xmr_required_payment_id_short": "3caca930a471a739",
    "created_at": "2017-07-01T08:11:27Z"
    "expires_at": "2017-07-01T08:26:27Z",
    "seconds_till_timeout": 857,
    "xmr_amount_total": 5.68,
    "xmr_amount_remaining": 5.68,
    "xmr_num_confirmations_remaining": -1,
    "xmr_recommended_mixin": 4,
    "btc_num_confirmations_before_purge": 144,
    "btc_num_confirmations": 0,
    "btc_transaction_id": ""
}
```


In this example, the next step would require the user to pay *5.68* XMR to the (integrated) Monero address *4EAAQR...*.

As of September 2019, instead of paying to the integrated Monero address, the user can also pay to the provided Monero subaddress.

In case the user's wallet does not support integrated addresses, the user can pay via the old-style address *44TVPc...* while providing the (long) payment ID *356620...*.

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

Note: The field *xmr_required_amount* is deprecated in favor of *xmr_amount_total*. The field *xmr_receiving_integrated_address* is deprecated in favor of *xmr_receiving_subaddress*. The field *xmr_required_payment_id_long* is deprecated in favor of *xmr_receiving_subaddress*. The field

xmr_required_payment_id_short is deprecated in favor of *xmr_receiving_subaddress*. The field *xmr_receiving_address* is deprecated in favor of *xmr_receiving_subaddress*.

1.2.6 Querying order price

API endpoint: https://xmr.to/api/v2/xmr2btc/order_check_price/

The order status endpoint allows users to query the recent price of an order.

Request

Issue a *POST* request to query the price of a given order. You have to supply the amount of BTC `btc_amount` in the request:

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

Response

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

```
{
    "btc_amount": <requested_amount_in_btc_as_float>,
    "xmr_amount_total": <amount_in_xmr_for_this_order_as_float>,
    "xmr_num_confirmations_remaining": <num_xmr_confirmations_remaining_before_
↪bitcoins_will_be_sent_as_integer>,
    "xmr_price_btc": <price_of_1_btc_in_xmr_as_offered_by_service_as_float>
}
```

Errors

On failure, one of the following errors is returned:

- `XMRTO-ERROR-004`
- `XMRTO-ERROR-005`
- `XMRTO-ERROR-009`
- `XMRTO-ERROR-012`
- `XMRTO-ERROR-014`

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

Rate limitation

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

Example

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

Request

```
curl --data '{"btc_amount": "0.15"}' -H "Content-Type: application/json" \
    https://xmr.to/api/v2/xmr2btc/order_check_price/
```

or

```
http --json https://xmr.to/api/v2/xmr2btc/order_check_price/ btc_amount=0.15
```

Response

The response gives the current price for the order:

```
{
    "btc_amount": 0.15,
    "xmr_amount_total": 11.5296804,
    "xmr_num_confirmations_remaining": 1,
    "xmr_price_btc": 0.0130099
}
```

In this example, the order including the payment of 0.15 BTC would require the user to pay *11.5296804* XMR.

1.3 Version 1

Version 1 was the original version of the XMR.to API. Active from 2015 till July 2017.

Version 1 supports Monero subaddresses.

Note: API version 1 is deprecated since July 2017.

1.3.1 Changelog

Date	Change
Sep 2019	Added xmr_receiving_subaddress field to response of `order_status_query`.

1.3.2 Querying order parameters

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_float>,
    "price": <price_of_1_btc_in_xmr_as_offered_by_service_as_float>,
    "upper_limit": <upper_order_limit_in_btc_as_float>
}
```

Fields should be self-explanatory.

Errors

On failure, one of the following errors is returned:

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

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

Rate limitation

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

Example

Request

```
curl https://xmr.to/api/v1/xmr2btc/order_parameter_query/
```

or

```
http https://xmr.to/api/v1/xmr2btc/order_parameter_query/
```

Response

```
{
    "price": 0.004011,
    "upper_limit": 1.0,
    "lower_limit": 0.001
}
```

1.3.3 Creating a new order

API endpoint: https://xmr.to/api/v1/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 and amount to create the order.

Request

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

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

Note: Make sure that `btc_amount` 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_float>,
    "btc_dest_address": <requested_destination_address_as_string>,
    "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-002`
- `XMRTO-ERROR-003`
- `XMRTO-ERROR-004`
- `XMRTO-ERROR-005`
- `XMRTO-ERROR-014`

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

Rate limitation

It is possible to request the endpoint up to **4 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", \
    "btc_amount": 0.1}' -H "Content-Type: application/json" https://xmr.to/api/v1/
↪xmr2btc/order_create/
```

or

```
http --json https://xmr.to/api/v1/xmr2btc/order_create/ btc_dest_
↪address=1FhnVJi2V1k4MqXm2nHoEbY5LV7FPai7bb btc_amount=0.1
```

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

Response

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

1.3.4 Querying order status

API endpoint: https://xmr.to/api/v1/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_float>,
    "btc_dest_address": <requested_destination_address_as_string>,
    "uuid": <unique_order_identifier_as_12_character_string>
    "btc_num_confirmations": <btc_num_confirmations_as_integer>,
    "btc_num_confirmations_before_purge": <btc_num_confirmations_before_purge_as_
↪integer>,
    "btc_transaction_id": <btc_transaction_id_as_string>,
    "created_at": <timestamp_as_string>,
    "expires_at": <timestamp_as_string>,
    "seconds_till_timeout": <seconds_till_timeout_as_integer>,
```

(continues on next page)

(continued from previous page)

```
    "xmr_amount_total": <amount_in_xmr_for_this_order_as_float>,
    "xmr_amount_remaining": <amount_in_xmr_that_the_user_must_still_send_as_float>,
    "xmr_num_confirmations_remaining": <num_xmr_confirmations_remaining_before_
↪bitcoins_will_be_sent_as_integer>,
    "xmr_price_btc": <price_of_1_btc_in_xmr_as_offered_by_service_as_float>,
    "xmr_receiving_subaddress": <xmr_subaddress_user_needs_to_send_funds_to_as_string>
↪,
    "xmr_receiving_address": <xmr_address_user_needs_to_send_funds_to_as_string>,
    "xmr_required_amount": <xmr_amount_user_needs_to_send_as_float>,  (deprecated)
    "xmr_required_payment_id": <xmr_payment_id_user_needs_to_include_when_paying_as_
↪string>
}
```

Note: The field *xmr_required_amount* is deprecated in favor of *xmr_amount_total*. The field *xmr_receiving_integrated_address* is deprecated in favor of *xmr_receiving_subaddress*. The field *xmr_required_payment_id_long* is deprecated in favor of *xmr_receiving_subaddress*. The field *xmr_required_payment_id_short* is deprecated in favor of *xmr_receiving_subaddress*. The field *xmr_receiving_address* is deprecated in favor of *xmr_receiving_subaddress*.

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

Value	Meaning
`TO_BE_CREATED`	order creation pending
`UNPAID`	waiting for Monero payment by user
`UNDERPAID`	order partially paid
`PAID_UNCONFIRMED`	order paid, waiting for confirmation
`PAID`	order paid and confirmed
`BTC_SENT`	bitcoin payment sent
`TIMED_OUT`	order timed out before payment was complete
`NOT_FOUND`	order wasn't found in system (never existed or was purged)

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-014`

For error details see the *List of all errors 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/v1/xmr2btc/order_status_query/
```

or

```
http --json https://xmr.to/api/v1/xmr2btc/order_status_query/ uuid=xmrto-VkT2yM
```

Response

The response gives the current status of the order:


```
{
    "xmr_price_btc": 0.003963,
    "uuid": "xmrto-XCZEsu",
    "state_str": "UNPAID",
    "btc_amount": 0.1,
    "btc_dest_address": "1FhnVJi2V1k4MqXm2nHoEbY5LV7FPai7bb",
    "xmr_required_amount": 25.233409,
    "xmr_receiving_subaddress":
↪"83BGzCTthheE2KxNTBPnPJjJUthYPfDfCf3ENSVQcpga8RYSxNz9qCz1qp9MLye9euMjckGi11cRdeVGqsVqTLgH8w5fJ1D
↪",
    "xmr_receiving_address":
↪"44TVPcCSHebEQp4LnapPkhb2pondb2Ed7GJJLc6TkKwtSyumUnQ6QzkCCkojZycH2MRfLcujCM7QR1gdnRULRraV4UpB5n4
↪",
    "xmr_required_payment_id":
↪"223907873a29a00e3a5ff563c3b65f278ab6eb0cba623428ca3d9aaa54ea7bbb",
    "created_at": "2015-04-01T16:03:27Z",
    "expires_at": "2015-04-01T16:08:27Z",
    "seconds_till_timeout": 224,
    "xmr_amount_total": 25.233409,
    "xmr_amount_remaining": 25.233409,
    "xmr_num_confirmations_remaining": -1,
    "btc_num_confirmations_before_purge": 144,
    "btc_num_confirmations": 0,
    "btc_transaction_id": ""
}
```


In this example, the next step would require the user to pay *25.233409* XMR to the Monero address *44TV...B5n4* while providing the payment ID *2239...7bbb*.

As of September 2019, instead of paying to the Monero address, the user can also pay to the provided Monero subaddress.

Note: The payment **must** be made before the order expires, in this case, inside *224* seconds.

Note: The field *xmr_required_amount* is deprecated in favor of *xmr_amount_total*. The field *xmr_receiving_integrated_address* is deprecated in favor of *xmr_receiving_subaddress*. The field *xmr_required_payment_id_long* is deprecated in favor of *xmr_receiving_subaddress*. The field *xmr_required_payment_id_short* is deprecated in favor of *xmr_receiving_subaddress*. The field

xmr_receiving_address is deprecated in favor of *xmr_receiving_subaddress*.

1.4 Public test instance

XMR.to offers a public test instance to ease development on services depening on its API. The instance is connected to the Monero state 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

You can access the test instance through TOR using http://xmrtoozzngvdifcz.onion

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 2

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.