AttestationEndpoint

Endpoint address: /attestation
Available methods: POST, GET

This endpoint is responsible for handing all requests regarding attestation.

HTTP METHOD: GET

```
type=drop_identity
type=outstanding -> [(mid_b64, attribute_name)]
type=outstanding_verify -> [(mid_b64, attribute_name)]
type=verification_output -> {hash_b64: [(value_b64, match)]}
type=peers -> [mid_b64]
type=attributes&mid=mid_b64 -> [(attribute_name, attribute_hash)]
```

Arguments

Argument: type

Pattern	Annotations
`string (ascii)`	

Argument: mid

Pattern	Annotations
`string (base64)`	`The member id to use for 'attributes' type requests.`

Output

Return code: 200
Precondition: request.get('type', None) == 'attributes'

Pattern	Annotations
`[`	`Get the known attributes for a given member id (our own if no member id is supplied).`
`    [`	
`        string (ascii)`	`The attribute name.`
`        string (base64)`	`The attribute hash.`
`        {...}`	`The user metadata.`
`        string (base64)`	`The attester's member id.`
`    ]`	
`]`	

Return code: 200
Precondition: request.get('type', None) == 'verification_output'

Pattern	Annotations
`{`	`Poll available verification output.`
`    string (base64):`	
`        [`	
`            [`	
`                string (base64)`	`Attribute hash.`
`                number`	`Confidence in tested value.`
`            ]`	
`        ]`	
`}`	

Return code: 200
Precondition: request.get('type', None) == 'outstanding'

Pattern	Annotations
[	Poll outstanding attestation requests.
[	
string (base64)	The member id who requested attestation.
string (ascii)	The requested attribute name.
string (base64)	The JSON metadata as a string.
]	
]	

Return code: 200
Precondition: request.get('type', None) == 'drop_identity'

Pattern	Annotations
	Drop all identity data (used for debugging).

Return code: 200
Precondition: request.get('type', None) == 'peers'

Pattern	Annotations
[	Fetch all known overlay member identifiers.
string (base64)	
]	

Return code: 200
Precondition: request.get('type', None) == 'outstanding_verify'

Pattern	Annotations
[	Poll outstanding verification requests.
[	
string (base64)	The member id who requested attestation.
string (ascii)	The requested attribute name.
]	
]	

HTTP METHOD: POST

```
type=request&mid=mid_b64&attibute_name=attribute_name
type=allow_verify&mid=mid_b64&attibute_name=attribute_name
type=attest&mid=mid_b64&attribute_name=attribute_name&attribute_value=attribute_value_b64
type=verify&mid=mid_b64&attribute_hash=attribute_hash_b64&attribute_values=attribute_value_b64,...
```

Arguments

Argument: attribute_values

Pattern	Annotations
[	The values to match to/test for, when verifying.
string (base64)	
]	

Argument: mid

Pattern	Annotations
string (base64)	The member id to use for 'attributes' type requests.

Argument: attibute_name

Pattern	Annotations
string (ascii)	

Argument: attribute_value

Pattern	Annotations
string (base64)	

Argument: type

Pattern	Annotations
string (ascii)	

Argument: attribute_hash

Pattern	Annotations
string (base64)	The hash of the attribute to prove.

Output

Return code: 200
Precondition: request.get('type', None) == 'verify'

Pattern	Annotations
	Request verification of someone's attribute. Takes mid, attribute_values, attribute_hash and attribute_values.

Return code: 200
Precondition: request.get('type', None) == 'attest'

Pattern	Annotations
	Attest an attribute for someone. Takes mid, attribute_name and attribute_value.

Return code: 200
Precondition: request.get('type', None) == 'allow_verify'

Pattern	Annotations
	Allow verification of an attribute by someone. Takes mid and attribute_name.

Return code: 200
Precondition: request.get('type', None) == 'request'

Pattern	Annotations
	Request attestation for an attribute from an identifier. Takes mid and attribute_name.Optionally supply a custom metadata (string to string map) object.

DHTEndpoint

Endpoint address: /dht
Available methods:

This endpoint is responsible for handling requests for DHT data.

DHTPeersEndpoint

Endpoint address: /dht/peers
Available methods:

This endpoint is responsible for handling requests for DHT peers.

DHTStatisticsEndpoint

Endpoint address: /dht/statistics
Available methods: GET

This endpoint is responsible for returning statistics about the DHT.

HTTP METHOD: GET

Arguments

This endpoint method takes no arguments.

Output

Return code: 200

Pattern	Annotations
{	
"statistics":	
{	
"node_id":	
string (hex)	
"routing_table_buckets":	
number	
"num_keys_in_store":	
number	
"peer_id":	
string (hex)	
"num_tokens":	
number	
"routing_table_size":	
number	
"num_peers_in_store":	
{	
string (hex):	
number	
}	
"num_store_for_me":	
{	
string (hex):	
number	
}	
}	
}	

Return code: 404

Pattern	Annotations
{	
"error":	
string (ascii)	The available information in case of no success.
}	

DHTValuesEndpoint

Endpoint address: /dht/values
Available methods:

This endpoint is responsible for handling requests for DHT values.

NetworkEndpoint

Endpoint address: /network
Available methods: GET

This endpoint is responsible for handing all requests regarding the state of the network.

HTTP METHOD: GET

Arguments

This endpoint method takes no arguments.

Output

Return code: 200

Pattern	Annotations
{	All of the known peers and their services.
"peers":	
{	
string (base64):	The sha1 of the peer's public key.
{	
"services":	
[	
string (base64)	The sha1 of the Community's public key.
]	
"ip":	
string (base64)	
"port":	
number	
"public_key":	
string (base64)	
}	
}	
}	

OverlaysEndpoint

Endpoint address: /overlays
Available methods: GET

This endpoint is responsible for handing all requests regarding the status of overlays.

HTTP METHOD: GET

Arguments

This endpoint method takes no arguments.

Output

Return code: 200

Pattern	Annotations
{	All of the known overlays and their state.
"overlays":	
[	
{	
"global_time":	
number	
"peers":	
[	Pretty printed list of peers in this overlay.
string (ascii)	
]	
"master_peer":	
string (hex)	Public key of the overlay Community.
"my_peer":	
string (hex)	Public key of my member.
"overlay_name":	
string (ascii)	
}	
]	
}	

OverlayStatisticsEndpoint

Endpoint address: /overlays/statistics
Available methods: POST, GET

This endpoint is responsible for handing all requests regarding the statistics of overlays.

HTTP METHOD: GET

Arguments

This endpoint method takes no arguments.

Output

Return code: 200

Pattern	Annotations
{	The statistics per message per community, if available.
"statistics":	
[	
{	
string (ascii):	The Community name.
{	
string (ascii):	The message name.
{	
"bytes_down":	
number	
"bytes_up":	
number	
"last_measured_up":	
number	
"first_measured_up":	
number	
"num_down":	
number	
"identifier":	
number	The message number.
"last_measured_down":	
number	
"num_up":	
number	
"first_measured_down":	
number	
}	
}	
}	
]	
}	

HTTP METHOD: POST

```
.. http:post:: /overlays/statistics

A POST request to this endpoint will enable statistics on the given overlay.
- enable: whether to enable or disable the statistics (True/False)
- overlay_name: class name of the overlay
- all: if set to True, update applies to all overlays
```

```
**Example request**:

    .. sourcecode:: none

        curl -X PUT http://localhost:8085/ipv8/overlays/statistics
        --data "enable=True&overlay_name=overlay_name&all=True

**Example response**:

    .. sourcecode:: javascript

        {"success": True}
```

Arguments

Argument: all

Pattern	Annotations
boolean	Update applies to all overlays.

Argument: enable

Pattern	Annotations
boolean	Whether to enable or disable the statistics.

Argument: overlay_name

Pattern	Annotations
string (ascii)	Class name of the overlay.

Output

Return codes: 400, 412

Pattern	Annotations
{	
"success":	
boolean	Whether the request succeeded, see error.
"error":	
string (ascii)	The available information in case of no success.
}	

Return code: 200

Pattern	Annotations
{	
"success":	
boolean	Whether the request succeeded, see error.
}	

RootEndpoint

Endpoint address: /

Available methods:

The root endpoint of the HTTP API is the root resource in the request tree. It will dispatch requests regarding torrents, channels, settings etc to the right child endpoint.

SpecificDHTPeerEndpoint

Endpoint address: /dht/peers/%s
Available methods: GET

This class handles requests for a specific DHT peer.

HTTP METHOD: GET

Arguments

This endpoint method takes no arguments.

Output

Return code: 200

Pattern	Annotations
{	
"peers":	
{	
"public_key":	
string (base64)	
"address":	
[	
string (ascii)	
number	
]	
}	
}	

Return code: 500

Pattern	Annotations
{	
"error":	
{	
"message":	
string (ascii)	
"code":	
string (ascii)	
"handled":	
boolean	
}	
}	

Return code: 404

Pattern	Annotations
{	
"error":	
string (ascii)	The available information in case of no success.
}	

SpecificDHTValueEndpoint

Endpoint address: /dht/values/%s
Available methods: PUT, GET

This class handles requests for a specific DHT value.

HTTP METHOD: GET

Arguments

This endpoint method takes no arguments.

Output

Return code: 500

Pattern	Annotations
{	
"error":	
{	
"message":	
string (ascii)	
"code":	
string (ascii)	
"handled":	
boolean	
}	
}	

Return code: 404

Pattern	Annotations
{	
"error":	
string (ascii)	The available information in case of no success.
}	

Return code: 200

Pattern	Annotations
{	
"values":	
{	
"public_key":	
string (base64)	
"value":	
string (hex)	
}	
}	

HTTP METHOD: PUT

Arguments

Argument: value

Pattern	Annotations
string (hex)	The value to store.

Output

Return codes: 400, 404

Pattern	Annotations
{	
"error":	
string (ascii)	The available information in case of no success.
}	

Return code: 200

Pattern	Annotations
{	
"stored":	
boolean	
}	

Return code: 500

Pattern	Annotations
{	
"error":	
{	
"message":	
string (ascii)	
"code":	
string (ascii)	
"handled":	
boolean	
}	
}	

TrustchainBlocksEndpoint

Endpoint address: /trustchain/blocks

Available methods:

TrustchainEndpoint

Endpoint address: /trustchain
Available methods:

This endpoint is responsible for handing all requests regarding TrustChain.

TrustchainRecentEndpoint

Endpoint address: /trustchain/recent
Available methods: GET

HTTP METHOD: GET

Arguments

Argument: limit

Pattern	Annotations
number	

Argument: offset

Pattern	Annotations
number	

Output

Return code: 200

Pattern	Annotations
{	Fetch recently added blocks.
"blocks":	
[	
{	
"hash":	
string (hex)	
"timestamp":	
number	
"public_key":	
string (hex)	
"transaction":	
{...}	JSON object belonging to this type.
"type":	
string (ascii)	
"link_public_key":	
number	
"insert_time":	
string (ascii)	
"signature":	
string (hex)	
"previous_hash":	
string (hex)	
"link_sequence_number":	
number	
"sequence_number":	
number	
}	
]	
}	

TrustchainSpecificBlockEndpoint

Endpoint address: /trustchain/blocks/%s
Available methods: GET

HTTP METHOD: GET

Arguments

This endpoint method takes no arguments.

Output

Return code: 404

Pattern	Annotations
{	
"error":	
string (ascii)	The available information in case of no success.
}	

Return code: 200

Pattern	Annotations
{	Fetch a specific block.
"block":	
{	
"hash":	
string (hex)	
"timestamp":	
number	
"public_key":	
string (hex)	
"transaction":	
{...}	JSON object belonging to this type.
"type":	
string (ascii)	
"link_public_key":	
number	
"insert_time":	
string (ascii)	
"signature":	
string (hex)	
"previous_hash":	
string (hex)	
"link_sequence_number":	
number	
"sequence_number":	
number	
"linked":	
{	The dictionary describing the linked block
"hash":	
string (hex)	
"timestamp":	
number	
"public_key":	
string (hex)	
"transaction":	

{...}	JSON object belonging to this type.
"type":	
string (ascii)	
"link_public_key":	
number	
"insert_time":	
string (ascii)	
"signature":	
string (hex)	
"previous_hash":	
string (hex)	
"link_sequence_number":	
number	
"sequence_number":	
number	
}	
}	
}	

TrustchainSpecificUserBlocksEndpoint

Endpoint address: /trustchain/users/%s/blocks
Available methods: GET

HTTP METHOD: GET

Arguments

Argument: limit

Pattern	Annotations
number	

Output

Return code: 404

Pattern	Annotations
{	
"error":	
string (ascii)	The available information in case of no success.
}	

Return code: 200

Pattern	Annotations
{	Fetch the known users.
"blocks":	
[	
{	
"hash":	
string (hex)	
"timestamp":	
number	
"public_key":	
string (hex)	
"transaction":	
{...}	JSON object belonging to this type.
"type":	
string (ascii)	
"link_public_key":	
number	
"insert_time":	
string (ascii)	
"signature":	
string (hex)	
"previous_hash":	
string (hex)	
"link_sequence_number":	
number	
"sequence_number":	
number	
"linked":	
{	The dictionary describing the linked block
"hash":	
string (hex)	
"timestamp":	
number	

"public_key":	
string (hex)	
"transaction":	
{...}	JSON object belonging to this type.
"type":	
string (ascii)	
"link_public_key":	
number	
"insert_time":	
string (ascii)	
"signature":	
string (hex)	
"previous_hash":	
string (hex)	
"link_sequence_number":	
number	
"sequence_number":	
number	
}	
}	
]	
}	

TrustchainSpecificUserEndpoint

Endpoint address: /trustchain/users/%s

Available methods:

TrustchainUsersEndpoint

Endpoint address: /trustchain/users
Available methods: GET

HTTP METHOD: GET

Arguments

Argument: limit

Pattern	Annotations
number	

Output

Return code: 200

Pattern	Annotations
{	Fetch the known users.
"users":	
{	
"public_key":	
string (hex)	
"blocks":	
number	
}	
}	

TunnelCircuitsEndpoint

Endpoint address: /tunnel/circuits
Available methods: GET

This endpoint is responsible for returning circuit information from the TunnelCommunity.

HTTP METHOD: GET

Arguments

This endpoint method takes no arguments.

Output

Return code: 404

Pattern	Annotations
{	
"error":	
string (ascii)	The available information in case of no success.
}	

Return code: 200

Pattern	Annotations
{	
"circuits":	
{	
"circuit_id":	
number	
"bytes_down":	
number	
"state":	
string (ascii)	
"bytes_up":	
number	
"goal_hops":	
number	
"type":	
string (ascii)	
"actual_hops":	
number	
}	
}	

TunnelEndpoint

Endpoint address: /tunnel
Available methods:

This endpoint is responsible for handling requests for DHT data.

TunnelExitsEndpoint

Endpoint address: /tunnel/exits
Available methods: GET

This endpoint is responsible for returning exit socket information from the TunnelCommunity.

HTTP METHOD: GET

Arguments

This endpoint method takes no arguments.

Output

Return code: 404

Pattern	Annotations
{	
"error":	
string (ascii)	The available information in case of no success.
}	

Return code: 200

Pattern	Annotations
{	
"circuits":	
{	
"circuit_from":	
number	
"bytes_down":	
number	
"bytes_up":	
number	
"enabled":	
boolean	
}	
}	

TunnelRelaysEndpoint

Endpoint address: /tunnel/relays
Available methods: GET

This endpoint is responsible for returning relay information from the TunnelCommunity.

HTTP METHOD: GET

Arguments

This endpoint method takes no arguments.

Output

Return code: 404

Pattern	Annotations
{	
"error":	
string (ascii)	The available information in case of no success.
}	

Return code: 200

Pattern	Annotations
{	
"circuits":	
{	
"circuit_from":	
number	
"bytes_down":	
number	
"is_rendezvous":	
boolean	
"circuit_to":	
number	
"bytes_up":	
number	
}	
}	