Hermes Documentation

Release 0.7.25

Digant C Kasundra

March 02, 2016

Contents

Contents:

CHAPTER 1

Introduction

Hermes logs events, generates tasks, and tracks tasks in logical groups.

CHAPTER 2

Terminology

Rather than mimic the overloaded and overused terminology typically used, and in keeping with the Dropbox principal of "cupcake," Hermes adopts a more interesting language.

2.1 Events and Event Types

Events double as journal entries, logging system activities like server restarts, and requests for action, such as a need to restart or turn off a server.

As journal entries, events provide an audit trail and can potentially be used to track a range of activities. As request entries, events can initialize labors and subsequent events would close these labors.

Each event must be of a predefined event type. An event type consists of a category and state, the combination of which provides meaningful grouping and definition:

```
ID  CATEGORY            STATE
[1] system-reboot       required
[2] system-reboot       completed
[3] system-maintenance  required
[4] system-maintenance  ready
[5] system-maintenance  completed
```

Event types are often written simply as `category-state`, such as `system-reboot-required`.

An individual event entry consists of the event type, the host, and the time of occurrence.

2.2 Labors

Labors represent tasks that need to be performed or outstanding issues that need to be addressed for a host. All labors are created and closed as the result of events.

Labors are usually referred to by the event which triggered its creation, so a `system-reboot-required` event creates a `system-reboot-required` labor.

2.3 Fates

2.3.1 Basics

The fates define how labors are created and completed. A typical fate will specify which event type will result in the creation of a labor for the host, and which event type will close labors for a host.

```
[1] system-reboot-required => system-reboot-completed
```

2.3.2 Chained Fates

An `intermediate` flag in the definition of a fate indicates if the fate only applies to existing labors. This allows fates to be chained together to essentially create a workflow engine.

For example:

```
[1] system-maintenance-required => system-maintenance-ready
[2] system-maintenance-ready => system-maintenance-completed
```

(with the second fate being flagged as an intermediate) would essentially mean:

```
system-maintenance-required => system-maintenance-ready => system-maintenance-completed
```

In this example, an event of type `system-maintenance-ready` only creates a labor if an existing labor created by an event of type `system-maintenance-required` was present.

2.3.3 Choose Your Own Adventure

Fates can allow multiple ways to resolve a labor.

```
[1] puppet-restart-required => puppet-restart-completed
[2] puppet-restart-required => system-restart-completed
```

In this example, a labor created by the event `puppet-restart-required` can be completed by either a `puppet-restart-completed` event, or a `system-restart-completed` event.

2.4 Quests

Quests are collections of labors, making tracking and reporting of progress much easier.

For example, when a security fix is released that requires all web servers to be restarted, a quest can be created with a `system-restart-required` labor for all the hosts.

Quests will eventually contain information to outside references, such as Jira tickets.

CHAPTER 3

Status

Development can be tracked at GitHub and Travis_CI

CHAPTER 4

TODOS

4.1 Deletion Support

Currently, nothing can be deleted through the API or client. It would be nice to be able to delete event-types and fates.

CHAPTER 5

Configuration

```
# Format for logging output.
# See https://docs.python.org/2/library/logging.html#logrecord-attributes
# Type: str
log_format: "%(asctime)-15s\t%(levelname)s\t%(message)s"

# Number of worker processes to fork for receving requests. This option
# is mutually exclusive with debug.
# Type: int
num_processes: 1

# The port to listen to requests on.
# Type: int
port: 10901

# The address to bind to. By default it listens on all interfaces.
# Type: string
bind_address: "127.0.0.1"

# Passing debug option down tornado. Useful for development to
# automatically reload code.
# Type: bool
debug: true

# The domain name to append to user names if not specified
domain: "dropbox.com"

# Specifies whether to use XSRF headers/cookies for API calls. Default: true
# Type: bool
api_xsrf_enabled: false

# Takes a SqlAlchemy URL to the database. More details
# can be found at the following URL:
# http://docs.sqlalchemy.org/en/rel_0_9/core/engines.html#database-urls
#
# Type: str
database: "mysql://localhost:3306/emsdb?user=emsdb&passwd=testpw"

# The server to use to host queries
query_server: "http://localhost:5353/api/query"

# Slack integration (optional)
# slack_webhook:  "https://hooks.slack.com/services/"
# slack_proxyhost: "proxyserver:port"
```

```
# Email notifications
email_notifications: false
# email_sender_address: "hermes@localhost"

# Always send email notifications to this comma seperated list
# email_always_copy: "admin@company.com"

# This is the expiration (in seconds) of auth_tokens used for API calls
# Type: int
auth_token_expiry: 600

# Sentry DSN if using Sentry to log exceptions.
# sentry_dsn:

# Additional plugin directory (full path)
# plugin_dir:

# Specify the org identifier for FullStory integration
# fullstory_id:

# StrongPOC integration (optional)
# strongpoc_server:

# Specify the environment - dev is default, set to prod for production
# environment: "dev"

# if environment is dev, send emails to the following email address instead
# of actual recipients
# dev_email_recipient:
```

CHAPTER 6

API Documentation

Hermes is designed as an API first so anything possible in the Web UI or command line tools would be available here.

6.1 Authentication

Authentication is still in the works. Right now, Hermes API is expected to sit behing some kind of authenticating proxy.

6.2 Requests

In addition to the authentication header above all `POST`/`PUT` requests will be sent as json rather than form data and should include the header `Content-Type: application/json`

6.3 Responses

All responses will be in `JSON` format along with the header `Content-Type: application/json` set.

The `JSON` payload will be in one of two potential structures and will always contain a `status` field to distinguish between them. If the `status` field has a value of `"ok"` or `"created"`, then the request (or creation, respectively) was successful and the response will be available the remaining fields.

```
{
    "status": "ok",
    "id": 1,
    ...
}
```

If the `status` field has a value of `"error"` then the response failed in some way. You will have access to the error from the `error` field which will contain an error `code` and `message`.

```
{
    "status": "error",
    "error": {
        "code": 404,
        "message": "Resource not found."
    }
}
```

6.4 Pagination

Most, if not all, responses that return a list of resources will support pagination. If the `data` object on the response has a `total` attribute then the endpoint supports pagination. When making a request against this endpoint `limit` and `offset` query parameters are supported.

An example response for querying the `sites` endpoint might look like:

```
{
    "status": "ok",
    "hosts": [
        {
            "id": 1
            "hostname": "example",
            "href": "/api/v1/hostname/example",
        }
    ],
    "limit": 10,
    "offset": 0,
    "total": 1
}
```

CHAPTER 7

API Reference

GET /api/v1/hosts/?
Get all Hosts

Example Request:

```
GET /api/v1/hosts HTTP/1.1
Host: localhost
```

Example response:

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

{
    "status": "ok",
    "hosts": [
        {
            "id": 1,
            "href": "/api/v1/hosts/server1",
            "hostname": "server1",
        },
        ...
    ],
    "limit": 10,
    "offset": 0,
    "totalHosts": 1,
}
```

Query Parameters

- **hostname** (*string*) – (*optional*) Filter Hosts by hostname.
- **hostQuery** (*string*) – (*optional*) the query to send to the plugin to come up with the list of hostnames
- **limit** (*int*) – (*optional*) Limit result to N resources.
- **offset** (*int*) – (*optional*) Skip the first N resources.

Status Codes

- 200 OK – The request was successful.
- 401 Unauthorized – The request was made without being logged in.

POST /api/v1/hosts/?

Create a Host entry

Example Request:

```
POST /api/v1/hosts HTTP/1.1
Host: localhost
Content-Type: application/json
{
    "hostname": "example"
}
```

or:

```
{
    "hosts": [
        {
            "hostname": "server1"
        },
        {
            "hostname": "server2"
        },
        ...
    ]
}
```

Example response:

```
HTTP/1.1 201 OK
Location: /api/v1/hosts/example

{
    "status": "created",
    "href": "/api/v1/hosts/example",
    "id": 1,
    "hostname": "example"
}
```

or:

Request JSON Object

- **hostname** (*string*) – The hostname of the server

Request Headers

- Content-Type – The server expects a json body specified with this header.

Response Headers

- Location – URL to the created resource.

Status Codes

- 201 Created – The Host was successfully created.
- 400 Bad Request – The request was malformed.
- 401 Unauthorized – The request was made without being logged in.
- 409 Conflict – There was a conflict with another resource.

GET /api/v1/hosts/(?P<hostname>.*)/?

Get a specific Host

Example Request:

```
GET /api/v1/hosts/example HTTP/1.1
Host: localhost
```

Example response:

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

{
    "status": "ok",
    "id": 1,
    "hostname": "example",
    "labors": [],
    "quests": [],
    "events": [],
    "href": "/api/v1/hosts/example",
    "limit": 10,
    "offset": 0,
    "lastEvent": "2015-05-05 22:13:11"
}
```

Parameters

- **hostname** (*string*) – hostname of the Host to retrieve

Query Parameters

- **expand** (*string*) – (*optional*) supports labors, events, eventtypes, quests
- **limit** (*int*) – (*optional*) Limit result of child resources.
- **offset** (*int*) – (*optional*) Skip the first N child resources.

Status Codes

- 200 OK – The request was successful.
- 401 Unauthorized – The request was made without being logged in.
- 404 Not Found – The Host was not found.

DELETE /api/v1/hosts/(?P<hostname>.*)/?

Delete a Host

Not supported

PUT /api/v1/hosts/(?P<hostname>.*)/?

Update a Host

Example Request:

```
PUT /api/v1/hosts/example HTTP/1.1
Host: localhost
Content-Type: application/json

{
    "hostname": "newname",
}
```

Example response:

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

{
    "status": "ok",
    "id": 1,
    "href": "/api/v1/hosts/example",
    "hostname": "newname",
}
```

Parameters

- **hostname** (*string*) – hostname of the Host that should be updated.

Request JSON Object

- **hostname** (*string*) – The new hostname of the Host.

Request Headers

- Content-Type – The server expects a json body specified with this header.

Status Codes

- 200 OK – The request was successful.
- 400 Bad Request – The request was malformed.
- 401 Unauthorized – The request was made without being logged in.
- 403 Forbidden – The request was made with insufficient permissions.
- 404 Not Found – The Host at hostname was not found.
- 409 Conflict – There was a conflict with another resource.

GET /api/v1/eventtypes/?
Get all EventTypes

Example Request:

```
GET /api/v1/eventtypes HTTP/1.1
Host: localhost
```

Example response:

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

{
    "status": "ok",
    "limit": 10,
    "offset": 0,
    "totalEventTypes": 3,
    "eventTypes": [
        {
            "id": 1,
            "category": "foo",
            "state": "bar",
            "description": "Foo bar all the way",
            "href": "/api/v1/eventtypes/1"
        },
        ...
```

```
    ],
}
```

Query Parameters

- **category** (*string*) – (*optional*) Filter EventTypes by category.
- **state** (*string*) – (*optional*) Filter EventTypes by state.
- **limit** (*int*) – (*optional*) Limit result to N resources.
- **offset** (*int*) – (*optional*) Skip the first N resources.
- **startingTypes** (*boolean*) – (*optional*) Return the event types that can create non-intermediate Labors

Status Codes

- 200 OK – The request was successful.
- 401 Unauthorized – The request was made without being logged in.

`POST /api/v1/eventtypes/?`

Create a EventType entry

Example Request:

```
POST /api/v1/eventtypes HTTP/1.1
Host: localhost
Content-Type: application/json
{
    "category": "system-reboot",
    "state": "required",
    "description": "System requires a reboot.",
}
```

or:

```
{
    "eventTypes": [
        {
            "category": "foo",
            "state": "bar",
            "description": "Some description"

        },
        {
            "category": "foo",
            "state": "baz",
            "description": "Some description",
             "restricted": true,
        },
        {
            "category": "tango",
            "state": "foxtrot",
            "description": "Some description"
        }
    ]
}
```

Example response:

```
HTTP/1.1 201 OK
Location: /api/v1/eventtypes/1

{
    "status": "created",
    "id": 1,
    "category": "system-reboot",
    "state": "required",
    "description": "System requires a reboot.",
    "restricted": false,
}
```

or:

```
{
    "status": "created",
    "eventTypes":
    [
        {
            "category": "foo",
            "state": "bar",
            "href": "/api/v1/eventtypes/7",
            "id": 7,
            "description": "Some description",
            "restricted": false,
        },
        {
            "category": "foo",
            "state": "baz",
            "href": "/api/v1/eventtypes/8",
            "id": 8,
            "description": "Some description",
            "restricted": true,
        },
        {
            "category": "tango",
            "state": "foxtrot",
            "href": "/api/v1/eventtypes/9",
            "id": 9,
            "description": "Some description",
            "restricted": false,
        }
    ],
    "totalEventTypes": 3
}
```

Request JSON Object

- **category** (*string*) – The category value of the EventType

Regjson string state The state value of the EventType

Regjson string description The human readable description of the EventType

Regjson boolean restricted (*optional*) If true, the EventType created will be restricted such that only direct API calls can throw events of that type (and the CLI/WebGUI would refuse)

Request Headers

- Content-Type – The server expects a json body specified with this header.

Response Headers

- Location – URL to the created resource.

Status Codes

- 201 Created – The site was successfully created.
- 400 Bad Request – The request was malformed.
- 401 Unauthorized – The request was made without being logged in.
- 409 Conflict – There was a conflict with another resource.

GET /api/v1/eventtypes/(?P<id>d+)/?
Get a specific EventType

Example Request:

```
GET /api/v1/eventtypes/1/ HTTP/1.1
Host: localhost
```

Example response:

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

{
    "status": "ok",
    "id": 1,
    "category": "system-reboot",
    "state": "required",
    "description": "This system requires a reboot",
    "events": [],
    "autoCreates": [],
    "autoCompletes": []
    "limit": 10,
    "offset": 0
}
```

Parameters

- **id** (*int*) – id of the EventType to retrieve

Query Parameters

- **expand** (*string*) – (*optional*) supports events, fates
- **limit** (*int*) – (*optional*) Limit result of child resources.
- **offset** (*int*) – (*optional*) Skip the first N child resources.

Status Codes

- 200 OK – The request was successful.
- 401 Unauthorized – The request was made without being logged in.
- 404 Not Found – The EventType was not found.

DELETE /api/v1/eventtypes/(?P<id>d+)/?
Delete an EventType

Not supported

PUT /api/v1/eventtypes/(?P<id>d+)/?
Update an EventType

Example Request:

```
PUT /api/v1/eventtypes/1/ HTTP/1.1
Host: localhost
Content-Type: application/json

{
    "description": "New description",
}
```

Example response:

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

{
    "status": "ok",
    "id": 1,
    "href": "/api/v1/eventtypes/1",
    "category": "system-reboot",
    "state": "required",
    "description": "New description",
}
```

Parameters

- **id** (*string*) – id of the EventType that should be updated.

Request JSON Object

- **description** (*string*) – The new description of the EventType.

Request Headers

- Content-Type – The server expects a json body specified with this header.

Status Codes

- 200 OK – The request was successful.
- 400 Bad Request – The request was malformed.
- 401 Unauthorized – The request was made without being logged in.
- 403 Forbidden – The request was made with insufficient permissions.
- 404 Not Found – The EventType was not found.
- 409 Conflict – There was a conflict with another resource.

GET /api/v1/events/?
Get all Events

Example Request:

Example response:

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

{
```

```
        "status": "ok",
        "limit": 10,
        "offset": 0,
        "totalEvents": 10,
        "events": [
            {
                "id": 1,
                "hostId": 1,
                "timestamp": "2015-06-01 12:11:01",
                "user": "jonny",
                "eventTypeId": 1,
                "note": "Event note",
            },
            ...
        ],
    }
```

Query Parameters

- **eventTypeId** (*int*) – (*optional/multiple*) Filter Events by EventType id.
- **hostId** (*int*) – (*optional*) Filter Events by Host id.
- **hostname** (*string*) – (*optional*) Filter Events by Host's hostname
- **limit** (*int*) – (*optional*) Limit result to N resources.
- **offset** (*int*) – (*optional*) Skip the first N resources.
- **after** (*string*) – (*optional*) Only select events at and after a given timestamp
- **before** (*string*) – (*optional*) Only select events before a given timestamp
- **afterEventType** (*int*) – (*optional*) Only select events at and after the last event of a given event type
- **hostQuery** (*string*) – (*optional*) Only select events that match a given host query

Status Codes

- 200 OK – The request was successful.
- 401 Unauthorized – The request was made without being logged in.

POST /api/v1/events/?

Create an Event entry

Example Request:

```
POST /api/v1/events HTTP/1.1
Host: localhost
Content-Type: application/json
{
    "hostname": "example",
    "user": "johnny",
    "eventTypeId": 3,
    "note": "Sample description"
}
```

or

```
POST /api/v1/events HTTP/1.1
Host: localhost
Content-Type: application/json
{
    "hostname": "example",
    "user": "johnny",
    "category": "system-reboot",
    "state": "completed",
    "note": "Sample description"
}
```

or

```
POST /api/v1/events HTTP/1.1
Host: localhost
Content-Type: application/json
{
    "hostQuery": "tag=value",
    "user": "johnny",
    "eventTypeId": 3,
    "note": "Sample description"
}
```

or

```
POST /api/v1/events HTTP/1.1
Host: localhost
Content-Type: application/json
{
    "questId": 1,
    "user": "johnny",
    "eventTypeId": 3,
    "note": "Sample description"
}
```

Example response:

```
HTTP/1.1 201 OK
Location: /api/v1/events/1

{
    "status": "created",
    "id": 1,
    "href": "/api/v1/events/1",
    "hostname": "example",
    "user": "johnny",
    "eventTypeId": 3,
    "note": "Sample description"
}
```

or

HTTP/1.1 201 OK Location: /api/v1/events/1

{ “status”: “created”, “events”: [{

“id”: 1, “href”: “/api/v1/events/1”, “hostname”: “example”, “user”: “johnny”, “eventTypeId”: 3, “note”: “Sample description”

```
    ]
}
```

Request JSON Object

- **hostname** (*string*) – (*optional*) The hostname of the Host of this Event
- **hostQuery** (*string*) – (*optional*) The external query to run to get Hosts for which to create Events

Regjson string hostnames (*optional*) The list of hostnames for which we want to throw this Event

Regjson int questId (*optional*) The Quest ID which has hosts for which we want to create Events

Regjson string user The user responsible for throwing this Event

Regjson int eventTypeId The id of the EventType

Regjson string category the category to use for the event

Regjson string state the state to use for the event

Regjson string note (*optional*) The human readable note describing this Event

Request Headers

- Content-Type – The server expects a json body specified with this header.

Response Headers

- Location – URL to the created resource.

Status Codes

- 201 Created – The Event was successfully created.
- 400 Bad Request – The request was malformed.
- 401 Unauthorized – The request was made without being logged in.
- 409 Conflict – There was a conflict with another resource.

GET /api/v1/events/(?P<id>d+)/?
Get a specific Event

Example Request:

```
GET /api/v1/events/1/ HTTP/1.1
Host: localhost
```

Example response:

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

{
    "status": "ok",
    "id": 1,
    "hostId": 1,
    "timestamp": "2015-06-01 12:11:01",
    "user": "jonny",
    "eventTypeId": 1,
    "note": "Event note",
}
```

Parameters

- **id** (*int*) – id of the Event to retrieve

Query Parameters

- **expand** (*string*) – (*optional*) supports hosts, eventtypes

Status Codes

- 200 OK – The request was successful.
- 401 Unauthorized – The request was made without being logged in.
- 404 Not Found – The EventType was not found.

DELETE /api/v1/events/(?P<id>d+)/?
Delete an Event

Not supported

PUT /api/v1/events/(?P<id>d+)/?
Update an Event

Not supported

GET /api/v1/fates/?
Get all Fates

Example Request:

```
GET /api/v1/fates HTTP/1.1
Host: localhost
```

Example response:

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

{
    "limit": 10,
    "offset": 0,
    "totalFates": 3,
    "fates": [
        {
            "id": 1,
            "href": "/api/v1/fates/1",
            "creationEventTypeId": 1,
            "followsId": null,
            "precedesIds": [],
            "forCreator": 0,
            "precedesIds": [3, 5],
            "description": "This is a fate",
        },
        ...
    ],
}
```

Query Parameters

- **limit** (*int*) – (*optional*) Limit result to N resources.
- **offset** (*int*) – (*optional*) Skip the first N resources.

- **expand** (*string*) – (*optional*) supports eventtypes

Status Codes

- 200 OK – The request was successful.
- 401 Unauthorized – The request was made without being logged in.

POST /api/v1/fates/?

Create a Fate entry

Example Request:

```
POST /api/v1/fates HTTP/1.1
Host: localhost
Content-Type: application/json
{
    "creationEventTypeId": 1,
    "description": "This is a fate",
    "followsId": 1,
    "forCreator": true,
}
```

Example response:

```
HTTP/1.1 201 OK
Location: /api/v1/fates/1

{
    "status": "created",
    "href": "/api/v1/fates/3",
    "id": 3,
    "creationEventTypeId": 1,
    "followsId": 1,
    "precedesIds": [],
    "forCreator": true,
    "description": "This is a fate"
}
```

Request JSON Object

- **creationEventTypeId** (*int*) – the ID of the EventType that triggers this Fate

Regjson int follows (*optional*) The ID of the Fate this Fate must come after, or null

Regjson string description (*optional*) The human readable description this Fate

Regjson boolean forOwner (*optional*) Indicates that Labors created by this Fate would be designated for action by the server owner. Default: true

Regjson boolean forCreator (*optional*) Indicates that Labors created by this Fate would be designated for action by the Quest owner. Default: false

Request Headers

- Content-Type – The server expects a json body specified with this header.

Response Headers

- Location – URL to the created resource.

Status Codes

- 201 Created – The Fate was successfully created.

- 400 Bad Request – The request was malformed.
- 401 Unauthorized – The request was made without being logged in.
- 409 Conflict – There was a conflict with another resource.

GET /api/v1/fates/(?P<id>d+)/?
Get a specific Fate

Example Request:

```
GET /api/v1/fates/1/ HTTP/1.1
Host: localhost
```

Example response:

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

{
    "status": "ok",
    "id": 1,
    "href": "/api/v1/fates/1",
    "creationEventTypeId": 1,
    "followsId": null,
    "precedesIds": [],
    "forCreator": false,
    "forOwner": true,
    "description": string,
}
```

Parameters

- **id** (*int*) – id of the Fate to retrieve

Query Parameters

- **expand** (*string*) – (*optional*) supports eventtypes

Status Codes

- 200 OK – The request was successful.
- 401 Unauthorized – The request was made without being logged in.
- 404 Not Found – The Fate was not found.

DELETE /api/v1/fates/(?P<id>d+)/?
Delete a Fate

Not supported

PUT /api/v1/fates/(?P<id>d+)/?
Update a Fate

Example Request:

```
PUT /api/v1/fates/3 HTTP/1.1
Host: localhost
Content-Type: application/json

{
    "description": "New desc",
```

```
    "followsId": 1
}
```

Example response:

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

{
    "status": "ok",
    "id": 3,
    "href": "/api/v1/fates/3",
    "creationEventTypeId": 1,
    "followsId": 1,
    "precedesId": [],
    "forCreator": false,
    "forOwner": true
    "description": "New desc"
}
```

Parameters

- **id** (*string*) – id of the Fate that should be updated.

Request JSON Object

- **description** (*string*) – The new description of this Fate.
- **intermediate** (*boolean*) – The new intermediate flag value.

Request Headers

- Content-Type – The server expects a json body specified with this header.

Status Codes

- 200 OK – The request was successful.
- 400 Bad Request – The request was malformed.
- 401 Unauthorized – The request was made without being logged in.
- 403 Forbidden – The request was made with insufficient permissions.
- 404 Not Found – The Fate was not found.
- 409 Conflict – There was a conflict with another resource.

GET /api/v1/labors/?
Get all Labors

Example Request:

```
GET /api/v1/labors HTTP/1.1
Host: localhost
```

Example response:

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

{
    "status": "ok",
```

```
        "limit": int,
        "offset": int,
        "totalFates": int,
        "labors": [
            {
                "id": 23,
                "startingLaborId": null,
                "href": "/api/v1/labors/23",
                "for_owner": false,
                "for_creator": true,
                "questId": 5,
                "hostId": 26,
                "creationTime": timestamp,
                "ackTime": timestamp,
                "targetTime": timestamp
                "ackUser": string,
                "completionTime": timestamp,
                "creationEventId": 127,
                "completionEventId": 212,
            },
            ...
        ],
    }
```

Query Parameters

- **hostname** (*string*) – (*optional*) filter Labors by a particular hostname
- **startingLaborId** (*string*) – (*optional*) get Labors by the Id or the Id of the starting labor
- **hostQuery** (*string*) – (*optional*) the query to send to the plugin to come up with the list of hostnames
- **userQuery** (*string*) – (*optional*) get labors for machines owned by this user or for which this user is responsible
- **category** (*string*) – (*optional*) limit labors to ones where the starting event type is of this category
- **state** (*string*) – (*optional*) limit labors to ones where the starting event type is of this state
- **open** (*boolean*) – if true, filter Labors to those still open
- **questId** (*int*) – the id of the quest we want to filter by
- **expand** (*string*) – (*optional*) supports hosts, eventtypes, events, quests
- **limit** (*int*) – (*optional*) Limit result to N resources.
- **offset** (*int*) – (*optional*) Skip the first N resources.

Status Codes

- 200 OK – The request was successful.
- 401 Unauthorized – The request was made without being logged in.

POST /api/v1/labors/?

Create a Labor entry

Not supported. Labors are only created by Fates.

GET /api/v1/labors/(?P<id>d+)/?
Get a specific Labor

Example Request:

```
GET /api/v1/labors/1 HTTP/1.1
Host: localhost
```

Example response:

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

{
    "status": "ok",
    "id": 23,
    "startingLaborId": null,
    "questId": 5,
    "hostId": 26,
    "for_creator": true,
    "for_owner": false,
    "creationTime": timestamp,
    "targetTime": timestamp,
    "ackTime": timestamp,
    "ackUser": string,
    "completionTime": timestamp,
    "creationEventId": 127,
    "completionEventId": 212,
}
```

Parameters

- **id** (*int*) – id of the Labor to retrieve

Query Parameters

- **expand** (*string*) – (*optional*) supports hosts, eventtypes

Status Codes

- 200 OK – The request was successful.
- 401 Unauthorized – The request was made without being logged in.
- 404 Not Found – The EventType was not found.

DELETE /api/v1/labors/(?P<id>d+)/?
Delete a Labor

Not supported

PUT /api/v1/labors/(?P<id>d+)/?
Update a Labor

Example Request:

```
PUT /api/v1/labors/23 HTTP/1.1
Host: localhost
Content-Type: application/json

{
    "questId": 1,
}
```

or

```
{
    "ackUser": "johnny"
}
```

Example response:

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

{
    "status": "ok",
    "id": 23,
    "questId": 1,
    "hostId": 26,
    "creationTime": timestamp,
    "targetTime": timestamp,
    "ackTime": timestamp,
    "ackUser": "johnny",
    "completionTime": timestamp,
    "creationEventId": 127,
    "completionEventId": 212,
}
```

Parameters

- **id** (*string*) – id of the Labor that should be updated.

Request JSON Object

- **questId** (*int*) – The Quest ID to which this Fate should now be associated.
- **ackUser** (*string*) – The username to log as having acknowledged this Labor

Request Headers

- Content-Type – The server expects a json body specified with this header.

Status Codes

- 200 OK – The request was successful.
- 400 Bad Request – The request was malformed.
- 401 Unauthorized – The request was made without being logged in.
- 403 Forbidden – The request was made with insufficient permissions.
- 404 Not Found – The Labor was not found.
- 409 Conflict – There was a conflict with another resource.

GET /api/v1/quests/?
Get all Quests

Example Request:

```
GET /api/v1/quests?progressInfo=true HTTP/1.1
Host: localhost
```

Example response:

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

{
    "status": "ok",
    "limit": int,
    "offset": int,
    "totalQuests": int,
    "quests": [
        {
            "id": 1,
            "href": "/api/v1/quests/1",
            "creator": "johnny",
            "embarkTime": timestamp,
            "targetTime": timestamp,
            "completionTime": timestamp,
            "description": "This is a quest almighty",
            "totalLabors": 20,
            "openLabors": 10,
            "percentComplete": 50,
            "labors": [],
        },
        ...
    ],
}
```

Query Parameters

- **filterClosed** (*boolean*) – (*optional*) if true, filter out completed Quests
- **progressInfo** (*boolean*) – (*optional*) if true, include progress information
- **byCreator** (*string*) – (*optional*) if set, filter the quests by a particular creator
- **hostnames** (*string*) – (*optional*) filter to quests that pertain to a particular host
- **hostQuery** (*string*) – (*optional*) filter quests to those involving hosts returned by the external query
- **limit** (*int*) – (*optional*) Limit result to N resources.
- **offset** (*int*) – (*optional*) Skip the first N resources.

Status Codes

- 200 OK – The request was successful.
- 401 Unauthorized – The request was made without being logged in.

POST /api/v1/quests/?

Create a Quest entry

Example Request:

```
POST /api/v1/quests HTTP/1.1
Host: localhost
Content-Type: application/json
{
    "fateId": 1,
    "creator": "johnny",
    "targetTime": timestamp,
    "description": "This is a quest almighty",
```

```
    "hostnames": [],
    "hostQuery": "tag=value"
}
```

Example response:

```
HTTP/1.1 201 OK
Location: /api/v1/hosts/example

{
    "status": "created",
    "id": 1,
    "href": "/api/v1/quests/1",
    "creator": "johnny",
    "embarkTime": timestamp,
    "targetTime": timestamp,
    "completionTime": timestamp,
    "description": "This is a quest almighty",
    "labors": [],
}
```

Request JSON Object

- **eventTypeId** (*int*) – the ID of the EventType to for the Events that will be thrown in the creation of this Quest

Regjson string creator the user creating this Quest

Regjson array hostnames the array of hostnames that will be part of this Quest

Regjson string hostQuery the query to send to the plugin to come up with the list of hostnames that will be part of this Quest

Regjson string description The human readable description this Quest

Regjson timestamp targetTime (*optional*) The target date for the completion of this Quest

Request Headers

- Content-Type – The server expects a json body specified with this header.

Response Headers

- Location – URL to the created resource.

Status Codes

- 201 Created – The Quest was successfully created.
- 400 Bad Request – The request was malformed.
- 401 Unauthorized – The request was made without being logged in.
- 409 Conflict – There was a conflict with another resource.

GET /api/v1/quests/(?P<id>d+)/?
Get a specific Quest

Example Request:

```
GET /api/v1/quests/1 HTTP/1.1
Host: localhost
```

Example response:

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

{
    "status": "ok",
    "id": 1,
    "href": "/api/v1/quests/1",
    "creator": "johnny",
    "embarkTime": timestamp,
    "targetTime": timestamp,
    "completionTime": timestamp,
    "description": "This is a quest almighty",
    "labors": [],
}
```

Parameters

- **id** (*int*) – id of the Quest to retrieve

Query Parameters

- **expand** (*string*) – (*optional*) supports labors, hosts, events, eventtypes
- **progressInfo** (*boolean*) – (*optional*) if true, include progress information
- **onlyOpenLabors** (*boolean*) – (*optional*) if true, only return open labors

Status Codes

- 200 OK – The request was successful.
- 401 Unauthorized – The request was made without being logged in.
- 404 Not Found – The EventType was not found.

DELETE /api/v1/quests/(?P<id>d+)/?

Delete a Quest

Not supported

PUT /api/v1/quests/(?P<id>d+)/?

Update a Quest

Example Request:

```
PUT /api/v1/quest/1 HTTP/1.1
Host: localhost
Content-Type: application/json

{
    "description": "New desc",
    "creator": "tammy"
}
```

Example response:

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

{
    "status": "ok",
    "id": 1,
```

```
    "href": "/api/v1/quests/1",
    "creator": "tammy",
    "embarkTime": timestamp,
    "targetTime": timestamp,
    "completionTime": timestamp,
    "description": "New desc",
    "labors": [],
}
```

Parameters

- **id** (*string*) – id of the Quest that should be updated.

Request JSON Object

- **description** (*string*) – the new description of the Quest
- **creator** (*string*) – The new username of the creator (owner)

Regjson timestamp targetTime Set a new targetTime

Request Headers

- Content-Type – The server expects a json body specified with this header.

Status Codes

- 200 OK – The request was successful.
- 400 Bad Request – The request was malformed.
- 401 Unauthorized – The request was made without being logged in.
- 403 Forbidden – The request was made with insufficient permissions.
- 404 Not Found – The Quest was not found.
- 409 Conflict – There was a conflict with another resource.

POST /api/v1/quests/(?P<id>d+)/mail/?

Send a message to all owners that are involved with a quest

Example Request:

```
POST /api/v1/quest/20/mail HTTP/1.1
Host: localhost
Content-Type: application/json
{
    "serverOwners": true,
    "laborOwners": false,
    "from": "user@example.com",
    "subject": "Hello!",
    "message": "Work is about to commence."
}
```

Example response:

```
HTTP/1.1 201 OK
Location: /api/v1/hosts/example

{
    "status": "created",
}
```

Note: Hermes will automatically append a link to the quest so users can go there directly from the email

:param id the ID of the quest we are working with when sending an email :regjson boolean serverOwners: send to all owners of servers that have labors in this quest :regjson boolean laborOwners: send to all labor owners (e.g. server owners if they own the active labor or the quest owner if they own the active labor) :regjson string from: the sender email address :regjson string subject: the subject line of the email :regjson string message: the body of the message

Request Headers

- Content-Type – The server expects a json body specified with this header.

Status Codes

- 201 Created – Email was created and sent

GET /api/v1/extquery/?
Get results from the external query services

The frontend will need to run queries against the external query server so that users can validate the results before working with a particular query. This handler acts as a passthrough so users can do exactly that.

Example Request:

```
GET /api/v1/query?query=server HTTP/1.1
Host: localhost
```

Example response:

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

{
    "status": "ok",
    "results": [
        {
            "id": 1,
            "href": "/api/v1/hosts/server1",
            "hostname": "server1",
        },
        ...
    ]
}
```

Status Codes

- 200 OK – The request was successful.
- 401 Unauthorized – The request was made without being logged in.

POST /api/v1/extquery/?
Pass through post to the external query handler

GET /api/v1/currentUser
Get a current authenticated user

Example Request:

```
GET /api/v1/currentUser HTTP/1.1
Host: localhost
```

Example response:

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

{
    "status": "ok",
    "user": "user@example.com"
}
```

Status Codes

- 200 OK – The request was successful.
- 401 Unauthorized – The request was made without being logged in.
- 403 Forbidden – The request was made with insufficient permissions.
- 404 Not Found – The User was not found.

GET /api/v1/serverConfig

Get the server's configuration information

This is used to get the config information that the front end might want to know about.

Example Request:

```
GET /api/v1/serverConfig HTTP/1.1
Host: localhost
```

Example response:

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

{
    "status": "ok",
    "domain": "example.com"
}
```

/api