Home

API Public Beta

Fever 1.14 introduces the new Fever API. This API is in public beta and currently supports basic syncing and consuming of content. A subsequent update will allow for adding, editing and deleting feeds and groups. The API's primary focus is maintaining a local cache of the data in a remote Fever installation.

I am soliciting feedback from interested developers and as such the beta API may expand to reflect that feedback. The current API is incomplete but stable. Existing features may be expanded on but will not be removed or modified. New features may be added.

I've created a simple HTML widget that allows you to query the Fever API and view the response.

Authentication

Without further ado, the Fever API endpoint URL looks like:

```
http://yourdomain.com/fever/?api
```

All requests must be authenticated with a `POST`ed `api_key`. The value of `api_key` should be the md5 checksum of the Fever accounts email address and password concatenated with a colon. An example of a valid value for `api_key` using PHP's native `md5()` function:

```
$email  = 'you@yourdomain.com';
$pass   = 'b3stp4s4wd3v4';
$api_key = md5($email.':'.$pass);
```

A user may specify that `https` be used to connect to their Fever installation for additional security but you should not assume that all Fever installations support `https`.

The default response is a JSON object containing two members:

`api_version` contains the version of the API responding

(positive integer)
`auth` whether the request was successfully authenticated (boolean integer)

The API can also return XML by passing `xml` as the optional value of the `api` argument like so:

```
http://yourdomain.com/fever/?api=xml
```

The top level XML element is named `response`.

The response to each successfully authenticated request will have `auth` set to `1` and include at least one additional member:

`last_refreshed_on_time` contains the time of the most recently refreshed (not *updated*) feed (Unix timestamp/integer)

Read

When reading from the Fever API you add arguments to the query string of the API endpoint URL. If you attempt to `POST` these arguments (and their optional values) Fever will not recognize the request.

Groups

```
http://yourdomain.com/fever/?api&groups
```

A request with the `groups` argument will return two additional members:

`groups` contains an array of `group` objects
`feeds_groups` contains an array of `feeds_group` objects

A `group` object has the following members:

`id` (positive integer)
`title` (utf-8 string)

The `feeds_group` object is documented under “Feeds/Groups Relationships.”

The “Kindling” super group is not included in this response and is composed of all feeds with an `is_spark` equal to `0`. The “Sparks” super group is not included in this response and is composed of all feeds with an `is_spark` equal to `1`.

Feeds

```
http://yourdomain.com/fever/?api&feeds
```

A request with the `feeds` argument will return two additional members:

- `feeds` contains an array of `group` objects
- `feeds_groups` contains an array of `feeds_group` objects

A `feed` object has the following members:

- `id` (positive integer)
- `favicon_id` (positive integer)
- `title` (utf-8 string)
- `url` (utf-8 string)
- `site_url` (utf-8 string)
- `is_spark` (boolean integer)
- `last_updated_on_time` (Unix timestamp/integer)

The `feeds_group` object is documented under “Feeds/Groups Relationships.”

The “All Items” super feed is not included in this response and is composed of all items from all feeds that belong to a given group. For the “Kindling” super group and all user created groups the items should be limited to feeds with an `is_spark` equal to `0`. For the “Sparks” super group the items should be limited to feeds with an `is_spark` equal to `1`.

Feeds/Groups Relationships

A request with either the `groups` or `feeds` arguments will return an additional member:

A `feeds_group` object has the following members:

- `group_id` (positive integer)
- `feed_ids` (string/comma-separated list of positive integers)

Favicons

```
http://yourdomain.com/fever/?api&favicons
```

A request with the `favicons` argument will return one additional member:

- `favicons` contains an array of `favicon` objects

A `favicon` object has the following members:

`id` (positive integer)
`data` (base64 encoded image data; prefixed by image type)

An example `data` value:

```
image/gif;base64,R0lGODlhAQABAIAAAObm5gAAACH5BAEAAAAALAAA
```

The `data` member of a `favicon` object can be used with the `data:` protocol to embed an image in CSS or HTML. A PHP/HTML example:

```
echo '<img src="data:'.$favicon['data'].'">';
```

Items

```
http://yourdomain.com/fever/?api&items
```

A request with the `items` argument will return two additional members:

`items` contains an array of `item` objects
`total_items` contains the total number of items stored in the database (added in API version 2)

An `item` object has the following members:

`id` (positive integer)
`feed_id` (positive integer)
`title` (utf-8 string)
`author` (utf-8 string)
`html` (utf-8 string)
`url` (utf-8 string)
`is_saved` (boolean integer)
`is_read` (boolean integer)
`created_on_time` (Unix timestamp/integer)

Most servers won't have enough memory allocated to PHP to dump all items at once. Three optional arguments control determine the items included in the response.

Use the `since_id` argument with the highest id of locally cached items to request 50 additional items. Repeat until the `items` array in the response is empty.

Use the `max_id` argument with the lowest id of locally cached items (or `0` initially) to request 50 previous items. Repeat until the `items` array in the response is empty. (added in API version 2)

Use the `with_ids` argument with a comma-separated list of item ids to request (a maximum of 50) specific items. (added in API version 2)

Hot Links

```
http://yourdomain.com/fever/?api&links
```

A request with the `links` argument will return one additional member:

`links` contains an array of `link` objects

A `link` object has the following members:

`id` (positive integer)
`feed_id` (positive integer) only use when `is_item` equals `1`
`item_id` (positive integer) only use when `is_item` equals `1`
`temperature` (positive float)
`is_item` (boolean integer)
`is_local` (boolean integer) used to determine if the source feed and favicon should be displayed
`is_saved` (boolean integer) only use when `is_item` equals `1`
`title` (utf-8 string)
`url` (utf-8 string)
`item_ids` (string/comma-separated list of positive integers)

When requesting hot links you can control the range and offset by specifying a length of days for each as well as a page to fetch additional hot links. A request with just the `links` argument is equivalent to:

```
http://yourdomain.com/fever/?api&links&offset=0&range=7&p
```

Or the first page (`page=1`) of Hot links for the past week (`range=7`) starting now (`offset=0`).

Link Caveats

Fever calculates Hot link temperatures in real-time. The API assumes you have an up-to-date local cache of items, feeds and favicons with which to construct a meaningful Hot view. Because they are ephemeral Hot links should not be cached in the same relational manner as items, feeds, groups and favicons.

Because Fever saves items and not individual links you can only "save" a Hot link when `is_item` equals `1`.

Sync

The `unread_item_ids` and `saved_item_ids` arguments can be used to keep your local cache synced with the remote Fever installation.

```
http://yourdomain.com/fever/?api&unread_item_ids
```

A request with the `unread_item_ids` argument will return one additional member:

> `unread_item_ids` (string/comma-separated list of positive integers)

```
http://yourdomain.com/fever/?api&saved_item_ids
```

A request with the `saved_item_ids` argument will return one additional member:

> `saved_item_ids` (string/comma-separated list of positive integers)

One of these members will be returned as appropriate when marking an item as read, unread, saved, or unsaved and when marking a feed or group as read.

Because groups and feeds will be limited in number compared to items, they should be synced by comparing an array of locally cached feed or group ids to an array of feed or group ids returned by their respective API request.

Write

The public beta of the API does not provide a way to add, edit or delete feeds or groups but you can mark items, feeds and groups as read and save or unsave items. You can also unread recently read items. When writing to the Fever API you add arguments to the `POST` data you submit to the API endpoint URL.

Adding `unread_recently_read=1` to your `POST` data will mark recently read items as unread.

You can update an individual item by adding the following three arguments to your `POST` data:

> `mark=item`
> `as=?` where `?` is replaced with `read`, `saved` or `unsaved`
> `id=?` where `?` is replaced with the `id` of the item to modify

Marking a feed or group as read is similar but requires one additional argument to prevent marking new, unreceived items as

read:

- `mark=?` where `?` is replaced with `feed` or `group`
- `as=read`
- `id=?` where `?` is replaced with the `id` of the feed or group to modify
- `before=?` where `?` is replaced with the Unix timestamp of the the local client's most recent `items` API request

You can mark the "Kindling" super group (and the "Sparks" super group) as read by adding the following four arguments to your `POST` data:

- `mark=group`
- `as=read`
- `id=0`
- `before=?` where `?` is replaced with the Unix timestamp of the the local client's last `items` API request

Similarly you can mark just the "Sparks" super group as read by adding the following four arguments to your `POST` data:

- `mark=group`
- `as=read`
- `id=-1`
- `before=?` where `?` is replaced with the Unix timestamp of the the local client's last `items` API request