Poesys Requirements

**Accounting API**

Robert J. Muller

June 30, 2018

36 Whitney Street  
San Francisco, CA 94131

# Introduction

The Accounting API is the REST application programming interface (API) for the Poesys accounting system. There are four services:

* **users:** requests relating to user registration, maintenance, and login
* **transactions:** requests relating to transactions, items, and reimbursements
* **entities:** requests relating to accounting entities, accounts, and fiscal-year accounts
* **fiscal\_years:** requests relating to fiscal years and fiscal-year accounts

Each request in the API starts with "https://www.poesys.com/api/". The request then includes the service name (transactions, for example). A URI with just the service name requests all the objects accessed by the service. So, for example, GET https://www.poesys.com/api/transactions retrieves one JSON object for each transaction in the API in a JSON array. The system reserves the right to limit retrieval for services that access very large numbers of objects.

The API supports the following HTTP verbs that take action against the database. Idempotence is the quality of getting the same result from multiple executions.

* **GET:** requests an object or array of objects identified by the URI endpoint
* **POST:** inserts an array of objects identified by the URI endpoint and sent as JSON content (not idempotent)
* **PUT:** updates an array of objects identified by the URI endpoint and sent as JSON content; replaces entire object, so it is idempotent (you can execute the PUT as many times as necessary and the result will be the same)
* **PATCH:** updates scalar values in the database based on existing objects and values (such as incrementing a counter in an object by one or replacing a value in an object rather than the whole object; not idempotent)
* **DELETE:** deletes an array of objects (not idempotent)

If the URI includes an identifier, the content returns a single object. Otherwise, the content objects include identifiers for each object in the content array.

GET https://www.poesys.com/api/transactions/45367/

gets a transaction object with transaction id 45367.

GET https://www.poesys.com/api/transactions?fiscal\_year=2017  
&account=Checking

gets an array of transaction objects against the account "Checking" in fiscal year 2017.

**Note: To make the documentation easier to read, the URIs as given do not include https://www.poesys.com/api/, which prefixes all URIs for this API.**

Each request may contain a set of parameters that qualify the request. A question mark (?) separates the parameters (the query) from the scheme, authority, and path. An ampersand (&) separates the individual parameters. Each parameter consists of the parameter name, an equal sign (=), and a text value. Quotes are taken literally and should be entered as HTTP codes along with other special characters for URIs. Dates should be in the following 24-hour format:

yyyy-MM-dd HH:mm:ss.SSS

For example, April 30, 2018 3pm would be

2018-04-30 15:00:00.000

The API uses JWT tokenization with cookies stored in the client. Always make the GET https://www.poesys.com/api/login request before attempting any other request to establish the JWT token in your client. All requests use SSL (HTTPS); you cannot use http:// to make an API request.

There is a special administrator privilege associated with certain users that permits certain requests such as /users. Only an administrator can change the administrator status of a user. The JWT token includes a flag for administrator privilege.

Each request has the following documentation:

request (the URI for the request; variables are in *italics*)

Parameters:

* a list of parameters with formatting constraints and \* meaning required

Actions:

* supported actions and what they do in the system (GET, PUT, and so on)
* standard response from each action, including JSON specification for content
* error responses from each action

Example:

one or more examples of potential requests

Notes:

* Any special issues with specific requests

***Note:*** *the API is not versioned and may change in the future; each request will document any backward-incompatible API changes, which are changes to existing functions that require a different structure (that is, a syntactic change to existing syntax) or that have different results (that is, a semantic change that makes results different than results from the same syntax in the previous version). New versions of this document will contain a section with all the backward-incompatible changes listed.*

# Users

The /users/ service performs all tasks associated with users (usernames, passwords, logins).

## Login

The /users/login request logs in your client to the API using the specified password. The response establishes a cookie with a JWT token in your client that will expire in 1 hour.

/users/*username/entity/*login

The username is a string with up to 100 characters and is required. The entity is the name of the accounting entity and is a string with up to 500 characters and is required. Note that you must submit all URIs with HTTPS, so all passwords are transmitted encrypted.

Parameters:

* **password:** a password string\*

Actions:

* **GET:** logs in the user with the specified username and entity using the password and returns a JWT login token; returns a different token for each call (that is, it does not "query" an existing token but always generates a new one); the token contains the entity name for use in all requests made with the token
  + Standard response 200: [{"token":"*token-value*"}]
  + Error response 400: invalid username, password, or entity
* **DELETE:** logs out the user with the specified username and entity name; returns a request to delete the JWT token from the client
  + Standard response 200: no content
  + Error response 400: invalid username, password, or entity

Example:

GET /users/randy/acme/login

Notes:

* Any request made with the JWT login token applies only to the entity specified in the login request. A user may access only a single accounting entity with a single login.

## Users

/users

The /users request gets a list of all registered users.

Actions:

* **GET:** queries the names of all registered users
  + Standard response 200: [{"user":"*name*"}, ...]

Example:

GET /users

Notes:

* Requires administrator privilege.

## Users/*username/entity*

/users/*username/entity*

The /users request processes information for a single user and accounting entity. Note that you cannot retrieve a password with this API request; GET verifies a password. The username is a required string with up to 100 characters, as is the entity. The names may contain encoded HTML characters that translate to ASCII characters in the database.

Parameters:

* **password:** a password string

Actions:

* **GET:** returns the username if the username is registered, the password is correct, and the user is associated with the accounting entity
* **POST:** registers the new user specified by the username with the given password and entity in the database; requires administrator privilege
* **PUT:** updates the user's password for the specified username; the user must be logged in with the username and entity or as a user with administrator privilege
* **DELETE:** removes the user specified by the username and entity from the database; the user must be logged in with the designated entity

Example:

POST /users/randy/acme?password=pw

Notes:

* POST requires administrator privilege. This restriction prevents open registration of users, which could be used for spamming or hacking.
* The system stores passwords as PBKDF2 hashes with the accompanying randomly generated salts. You cannot retrieve the original password; if you forget the original password the system must generate a new password for you.
* PUT allows updating a password to a new password by another user if that other user has administrator privilege. This allows the administrator to reset a password to a new one that they can send to the user.

# Transactions

The /transactions/ service handles all requests relating to transactions and their items, including reimbursements for receivables. See the /entities/ service for endpoints that manage entities and accounts. See the /fiscal\_years/ service for endpoints that manage fiscal years and their accounts.

## Transactions

The /transactions request manages a set of transactions. The content for PUT, POST, and DELETE contains JSON transaction object arrays. The JSON objects contain the relevant identifiers for existing objects.

/transactions?

Parameters:

* **description:** text describing the transaction (supports % and \_ wildcards)
* **account:** the name of an account belonging to the accounting entity that one of the items in the transaction is against
* **fiscal\_year:** the number of the fiscal year containing the transaction
* **start\_date:** the inclusive beginning of a date range that contains the transaction date; if there is an end date and you do not specify the start date, the query assumes a start date identical to the end date with hours/minutes/seconds set to 0 (start of day).
* **end\_date:** the inclusive end of a date range that contains the transaction date; if there is a start date and you do not specify the end date, the query assumes an end date identical to the start date with hours/minutes/seconds set to maximum values (end of day)
* **checked:** a boolean flag (true or false) indicating whether the transaction is reconciled
* **balance:** a boolean flag (true or false) indicating whether the transaction is a balance transaction, a transaction that has a single item that adds a balance to an account for the first time

Actions:

* **GET:** queries a set of transactions using any of the optional parameters to qualify the query; if you supply no parameters, returns all transactions in the database relating to the accounting entity over all fiscal years (HINT: don't do that without a really, really good reason.)
* **POST:** inserts the transactions (no parameters)
* **PUT:** updates the transactions (no parameters)
* **DELETE:** deletes the transactions (no parameters)

Examples:

GET /transactions?fiscal\_year=2017&entity=acme&account=Checking&checked=false  
GET /transactions?start\_date=2017-03-01  
POST /transactions

Notes:

The dates in the query may contain any level of information at the day interval level. Hours, minutes, and seconds not specified are taken as 0. If you do not specify a range end date, hours, minutes, and seconds for the default end date are the maximum values (end of the day). The description, entity, and account strings are case sensitive.

Account names are relative to the accounting entity, which you specify when you log in. Queries get the entity name from the JWT security token that login creates.