TODAQ

TODA-as-a-Service Developer Tutorial

The purpose of this tutorial is to familiarize development teams with the basic concepts in the TODA-as-a-Service API, and to exercise communication between your client code and TODAQ. This **step by step** guide will walk through each essential operation.

You will need:

- ✓ To have read and familiarized yourself with *The TODA Primer* and the TAAS API documentation at `https://docs.developer.todaqfinance.net`
- ✓ Your TAAS API key
- ✓ Your TAAS Dashboard login credentials

Managing Ownership

TODA lets you create assets (termed *files*), and enforce certain properties about their ownership. At the most basic level, TODA ensures that a file can never have more than one owner. More advanced applications can take advantage of a constraint system, where you are able to *encumber* the files you create so that they can only behave in a certain way[1].

[1] Encumbrances allow for advanced behaviour, such as restricting the ownership of a file to only the employees of a company, for example.

This tutorial describes the basic steps necessary to communicate over our standards-driven REST API.

Create Accounts

We need to create accounts in order to own assets on TODA. Every asset belongs to an account. Within your organization, you will likely create an account for every person, user, or other entity which will need to take ownership of a digital asset.

⇒ See the `Accounts - Create Individual` section of the API documentation

1. Following the API documentation, write a function to send a `POST` request to the *Create Individual* endpoint. While you can reference the example, ensure you replace the example API key with your own. If you encounter an HTTP error code, read the detailed response to ensure that you have provided valid inputs for each required field[2]. Upon a successful response, you will be provided with the Account ID for your newly-created account.

The API documentation includes examples that you can copy-paste directly into your command-line using `cURL`, but you can also select a different language to have it translate the examples into your favourite programming environment.

[2] The API documentation outlines which fields are *required* in each call.

2. ✓*Verify - API* At this time, it's important to verify that you can retrieve the account from our API. Following the examples in the

⇒ See the `Accounts - List` and `Accounts - Get By ID` sections of the API documentation

documentation, ensure that when you query for a list of accounts, you are able to retrieve the one you just created. Similarly, you should be able to retrieve it by providing the ID that was returned to you.

3. ✓*Verify - Portal* At this time, you should login to the portal at `https://developer.todaqfinance.net` using your credentials. The first thing you'll notice is that you're able to see a live view of your API requests[3]. If you change to the *Accounts* tab, you should be able to see and inspect the account you created. You'll notice that you can also use the portal to manually create accounts; this may be helpful during development.

[3] Keep the portal live event stream open during development to assist you in troubleshooting.

Create Files

⇒ See the `Files - Create` section of the API documentation

In order to transact assets, we need to create them first. A *file* in TODA is allowed to have any payload you wish. By default, the API supports storing the payload in JSON format, however storing the payload of a file is *not required* in TaaS[4].

[4] We have clients in sensitive fields who prefer not to store their file payloads on TaaS. Instead, they store the *hash* of their payload on TaaS. As you'll see, this is a supported option.

1. Create a file by sending a `POST` request to the *Create* endpoint for files, following the documentation. As noted, you may provide either the `payload` in JSON format, or a `payload-hash` as a 64-character hex string. You *will* need to specify an `initial-account` for the file.

2. Unlike the *Create Account* endpoint, this endpoint will not simply return a file identifier. Instead, it will return details about a *Transaction*. The transaction it returns will eventually contain the newly-created file. The transaction will initially have a status of `pending`, and will not include any files. As the transaction progresses, the file will be created and attached to the transaction, and the transaction will eventually complete, indicating that the `initial-account` has now received the file.

⇒ Watch the transaction creating the file progress in the live event stream in the portal. You should eventually see it reach a status of `recipient complete`.

3. The returned transaction has a nested property at `data.links.self`. This url represents the API call you can make to query the transaction in the future. Query this link now to see if the transaction has updated with more information. Eventually, the file will be attached to the transaction, listed under `data.relationships.files`.

4. Query the *Files - List* endpoint to see the full list of files in the organization. Additionally, experiment with querying *Files - Get By Id* and *Account - Get Files* to see how these return different subsets of files. Finally, note that your files now contain an `owner` property, reflecting the last-settled owner of the file.

File Types

The *Create File* endpoint accepts another relationship attribute, `file-type`. By setting the value of `data.relationships.file-type.data.id` to another file, you will create a file *of that type*. Any file can be used as the file type of another. For the moment, it's helpful to think of a file type as defining a way of categorizing your assets, such as *loyalty point*, or *shipping container*. As discussed, more advanced applications are able to take advantage of file type *encumbrances* to add additional behaviour to files of its type.

1. Take the opportunity to create two files which act as file types. In their payloads, specify one as being a *Gold Loyalty Point*, and another as a *Silver Loyalty Point*.
2. Create files of those types by specifying the file identifiers of either your gold or silver point as the `file-type` during the *File Create* call.
3. Fetch your newly created files using the *Files - Get* endpoint. You'll notice that full information about a file's type is included when the file is returned to you.

Bulk Creation

⇒ See the `Files - Bulk Create` section of the API documentation.

Often, clients prefer to create many files at once, all belonging to the same file type. In this case, you no longer specify the payload for the files (we auto-generate a serial number for you). Instead, you simply specify how many of the files to create.

1. Call the *Files - Bulk Create* endpoint, and create 100 files of type Silver.
2. Query the transaction until the file identifiers appear linked to the transaction. You'll be able to individually query each file at that time using *Files - Get By Id*.

Transactions

Creating files actually conducts a transaction to transfer the file to its initial owner, so we have already seen how transactions work. In the simplest case, simply specify which file identifiers you wish to transfer to another account:

1. Initiate a transaction by sending a `POST` request to the *Transactions - Initiate* endpoint. You will need to specify the `sender` and `recipient` as one of the accounts that have already been created. The `sender` account must be the current owner of the files you are sending.

2. As above, monitor this transaction to ensure it settles. Check the *Accounts - Get Files* endpoint to ensure that the files have moved from one account to another.

Bulk Transactions

Specifying individual files to transact is not always necessary. If you simply wish to transact some number of files of a certain type, this is easily supported by TaaS.

1. Bulk create 100 files of a certain type (i.e. *Silver*) if you have not done so already.
2. After the files have settled, transact 75 of them to another account, by using the *Transactions - Bulk Files of Type* endpoint. You will note both `file-type` and `quantity` are required fields.
3. Satisfy yourself that the transaction has properly settled by using the endpoints you are now familiar with.

Webhooks

Polling an API for updates is commonplace, but a more effective approach is for a *webhook* service to actively push events to an API client.

1. Login to the TaaS Dashboard using your credentials.
2. Click the *gear* icon in the upper-right to navigate to *Settings*.
3. Click *Enterprise* on the left-hand side to navigate to *Enterprise-wide Settings*.
4. Click *Add webhook* to enter a public URL for the API to notify with live updates.
5. Ensure your public URL is receiving notifications, by creating files and transacting them with the webhook enabled.

If your webhook is not receiving updates, check the live stream in the *Dashboard* tab, to ensure we are not generating errors attempting to connect to it.

For more information or assistance, please reach out to your Account Manager, or one of the following technical points of contact:

Adam Gravitis, Chief Technology Officer
adam.gravitis@todaqfinance.com

Dann Toliver, Co-Founder & Protocol Author
dann.toliver@todaqfinance.com