[

Address Book_banner

1920×768 118 KB

](https://europe1.discourse-cdn.com/business20/uploads/aave/original/2X/a/ab3d35722ede8941e6084c4f2b69a8373fe61866.jpeg)

TL;DR;

We have released the Aave Address Book, a library containing all the important addresses of the Aave ecosystem as easy-to-consume constants.

Current state

We have been supporting quite some governance proposal creations in the past and observed that there's a part that all these proposals have in common: they need to lookup addresses for the addresses provider they are targeting and either hardcode or fetch the oracle, configurator, … within their proposals.

This is a cumbersome task so we set out to ease the pain - which is where the Aave Address Book begins to shine

Address book

As most addresses we regularly work with are immutable for a market, it's only reasonable to persist them in a library so it's easily consumable by developers.

The library consists out of a generator that will generate market-specific libraries forevery registered market. The generator will also generate generic library allowing you to easily interact with multiple markets at once.

Market library

Market-specific libraries all follow a similar pattern and provide constants which developers can easily consume in their projects. They also provide a getToken(symbol)

function which will return a Token

struct containing the aToken, stableDebtToken, variableDebtToken and underlying token addresses.

library AaveV2Ethereum { ILendingPoolAddressesProvider internal constant POOL_ADDRESSES_PROVIDER = ILendingPoolAddressesProvider(0xB53C1a33016B2DC2fF3653530bfF1848a515c8c5);

```
ILendingPool internal constant POOL =
  ILendingPool(0x7d2768dE32b0b80b7a3454c06BdAc94A69DDc7A9);

ILendingPoolConfigurator internal constant POOL_CONFIGURATOR =
  ILendingPoolConfigurator(0x311Bb771e4F8952E6Da169b425E7e92d6Ac45756);

IAaveOracle internal constant ORACLE =
  IAaveOracle(0xA50ba011c48153De246E5192C8f9258A2ba79Ca9);

IAaveProtocolDataProvider internal constant AAVE_PROTOCOL_DATA_PROVIDER =
  IAaveProtocolDataProvider(0x057835Ad21a177dbdd3090bB1CAE03EaCF78Fc6d);

address internal constant POOL_ADMIN =
  0xEE56e2B3D491590B5b31738cC34d5232F378a8D5;

address internal constant EMERGENCY_ADMIN =
  0xCA76Ebd8617a03126B6FB84F9b1c1A0fB71C2633;

function getToken(string calldata symbol)
  public
  pure
  returns (Token memory m)
{
    ...
  }

}
```

Generic library

The generic library is best suited for tests & test helpers as using it might increase codeside and deployment cost. That said it behaves very similar to the Market libraries

with the difference of not exporting constants, but providing a getMarket(marketName)

and getToken(marketName, symbol)

getter.

Demo

To demonstrate usage we added the address book to thev2-token-listing-template which resulted in boilerplate reduction feat: address book by sakulstra · Pull Request #1 · bgd-labs/example-aave-v2-listing · GitHuband even in small deployment cost reduction from 483918

to 480725

.

In the actual payload we've been using AaveV2Ethereum

to archive smallest possible overhead, while in the helpers we've been using the generic AaveAddressBookV2

entrypoint so they are easily reusable for any market & network.

Here's a excerpt to highlight how easy this library can be used:

```
Token memory ens = AaveV2Ethereum.getToken('ENS');
AaveV2Ethereum.POOL_CONFIGURATOR.enableBorrowingOnReserve(ens.underlyingAsset, false);
```

In these two lines we managed to disable borrowing on ENS

without looking up a single address.

Next steps

We are quite happy with what we have so far, but there's obviously more that can be abstracted and reused in the ecosystem.

Therefore we plan to release some higher-level helpers like the ones used inv2-token-listing-template to help with more complex things, especially on the testing side of things.

Links

Aave Address Book

: https://github.com/bgd-labs/aave-address-book

Usage example

: https://github.com/bgd-labs/example-aave-v2-listing