0 Introduction

0.a. Purpose of the Smart Contracts

0.a.1 Sponsorships

Sponsoring users is a way for BrightID to have a continuous stream of funding while allowing BrightID to remain a public good. It spreads the burden of funding BrightID to many participants.

To use BrightID's verification system, a user must be sponsored--which happens just once per user per lifetime. In the most common scenario, an application purchases a sponsorship on behalf of a user. Purchasing sponsorships is recorded on Ethereum, while assigning a sponsorship to a BrightID user is recorded on BrightID nodes.

0.a.2 Sponsorship Contexts

Every sponsorship has at most one context, which indicates which application controls it.

When a sponsorship is newly created, it has no context assigned to it. The purchaser of the sponsorship can call a function on the smart contract to assign it a context. To prevent abuse, the context can only be assigned once.

0.a.3 Subscriptions

In addition to direct sales of sponsorships, a limited number of *subscriptions* will be available to early supporters. A subscription produces new sponsorships every month for nearly six years.

0.b Auditing Priorities

By far the most important contract to consider is the Sponsorships contract. It is designed to provide income for a long time. By contrast, subscriptions have a fixed lifetime and a cap. The Subscriptions contract is easier to fix if needed since the balances shouldn't change much--especially after the cap is reached.

The minter contracts (SponsorshipsMinter and SubscriptionsMinter) can be replaced using the functions of the MinterRole if issues are found after they have been deployed.

See also Handling Upgrades and Exploits to the Sponsorships Contract.

1 Sponsorships

1.a Name and Symbol

We will use the name **Sponsorships** to describe the asset (code). The symbol will be **Sp** (code).

1.b Transferability

Sponsorships are non-transferable (code, code).

1.c Balances

Sponsorships can be accumulated through purchase (code), but never depleted (because they inherit the NonTransferable module). The total balance should be visible from a wallet. The balance of how many sponsorships have been assigned to a context is visible from a web application we provide.

1.d Purchase

Purchases are handled using the SponsorshipsMinter contract (code).

Sponsorships are purchased for 1 DAI each (code, deployment script constants). Tokens used to purchase sponsorships are immediately sent to BMAIN DAO (code, code, code, code, code, code, deployment script constants).

Newly purchased sponsorships are not assigned a context (code).

1.d.1 Refunds

The number of sponsorships purchased is rounded down to the nearest unit and any payment left over is refunded (code).

Where possible, other tokens and ether sent to the smart contract are reclaimable.

1.d.2 Changing Purchase Token and Price

The contract owner of the SponsorshipsMinter can change the purchase token (code) (for example, from DAI to multi-collateral DAI) and price (code).

1.e Contexts

An address that holds sponsorships can have some that are assigned a context, and some that are unassigned (code).

An address can assign some number of their sponsorships to a given context (code).

1.f Balance Query Functions

There are functions to query the unassigned balance for an address (code), query the balance of any given context for any given address (code), and get the entire balance of a context across all addresses (code).

2 Subscriptions

2.a Name and Symbol

We will use the name **subscriptions** to describe the asset (code). The symbol will be **Subs** (code).

2.b Transferability

Subscriptions are non-transferable (code, code, code).

2.c Balances

Subscriptions can be accumulated through purchase (code), but never depleted (because they inherit the NonTransferable module). The total balance should be visible from MetaMask.

2.d Purchase

Purchases are handled using the SubscriptionsMinter contract (code).

2.d.1 Price Steps

The price increases in steps according to the following schedule. (code, code)

Step	Subscriptions	Price
1	400,000	1 DAI
2	500,000	2 DAI

2.d.2 Purchases spanning two steps

If a purchase would span two steps, only the subscriptions from the earlier step are purchased and the rest of the payment is refunded (code).

2.d.3 Refunds

The number of subscriptions purchased is rounded down to the nearest unit and any payment left over is refunded (code).

If a purchase is attempted after the sale has ended, the payment is refunded (code, code, code, code).

Where possible, other tokens and ether sent to the smart contract are reclaimable.

2.d.4 Direction of funds

Tokens used to purchase subscriptions are immediately sent to BMAIN DAO (code, code, code, code, code, code, deployment script constants).

2.d.5 Batches

Subscriptions can be purchased in batches, not just singly (code). For example, a batch of 1000 subscriptions could be purchased at the same time.

The starting timestamp[1] for each batch purchased must be recorded separately (code) and will be needed when sponsorships are claimed (code).

2.d.6 Claiming Sponsorships

New sponsorships are available from a subscription every month (starting with month zero--one sponsorship is claimable immediately), following this pattern:

Month (inclusive)	Sponsorships per month
0-11	1
12-23	2
24-35	3

[1] The timestamp doesn't have to be very precise. It could be months since October 1, 2019.

36-47	4
48-59	5
60-71	6

Months are actually thirty day periods. The final sponsorships can be claimed after 71 * 30 = 2130 days. One subscription produces 252 sponsorships in total. (Code.)

The SubscriptionsMinter claim function calls the corresponding claim function on the Subscriptions contract which returns the number of sponsorships which should be added to the balance for the calling address (code, code).

The SubscriptionsMinter contract then calls the Sponsorships contract's mint function which increments the addresses balance with new sponsorships accordingly (code).

When an address claims sponsorships, it claims all available sponsorships from all eligible batches (code).

3 Wallet Support (Asset Visibility)

3.a MetaMask

The total balance for both assets should be visible through MetaMask. This means the assets need to support the following:

balanceOf function ABI:

```
[{'constant': true, 'inputs': [{'name': '_owner', 'type': 'address'}], 'name': 'balanceOf',
'outputs': [{'name': 'balance', 'type': 'uint256'}], 'payable': false, 'type': 'function'}]
```

Public constants

```
string public constant symbol
uint8 public constant decimals
```

Applicable metamask code.

This is supported by Sponsorships (code, code) and Subscriptions (code, code, code). The NonTransferable contract from which they both inherit provides the balanceOf() function.

3.b Other Wallets

To be visible in certain other wallets also requires the presence of a

```
function totalSupply() public view returns (uint256) {
        return _totalSupply;
}
```

function that returns a value greater than zero. This relates to the internal `_mint` function and is supported by the NonTransferable contract (code) from which both Sponsorships (code) and Subscriptions (code, code) inherit.

4 Sending Tokens to BMAIN DAO

The Sponsorships (code), Subscriptions (code), SponsorshipsMinter (code), and SubscriptionsMinter (code) contracts all inherit from FinanceManager which allows tokens to be sent to a finance app using Aragon's deposit function definition. When used with an Aragon DAO, it records the transaction and deposits tokens in the vault. Transactions can have a reference telling what the transaction is for. Underlying _deposit function.

The SponsorshipsMinter and SubscriptionsMinter deposit tokens used for purchase, while all four contracts inheriting from FinanceManager allow any ERC20 tokens sent by mistake to be deposited.

The address of the finance app used by a contract can be changed by the owner (code).

5 Minters

The Sponsorships (code, code) and Subscriptions (code, code, code) contracts inherit OpenZeppelin's MinterRole. This means that the contract creator is designated as a Minter and Minters can add and remove other Minters. BMAIN DAO's Agent App should be the creator of both contracts so that it acquires the ability to replace minters. See deployment script.

5.a Minters for the Sponsorships contract

The SponsorshipsMinter, SubscriptionsMinter, and BMAIN DAO's Agent App are Minters for the Sponsorships contract. (See Initial Deployment.)

The SponsorshipsMinter needs to have the MinterRole to call the mint (code) function on the Sponsorships contract when someone wants to purchase sponsorships (code).

The SubscriptionsMinter needs to have the MinterRole to call the mint (code) function on the Sponsorships contract when someone wants to claim sponsorships from their subscriptions (code).

BMAIN DAO’s Agent App needs to have the MinterRole to replace the SponsorshipsMinter or SubscriptionsMinter contracts if needed. This also means that BMAIN DAO’s Agent App can mint sponsorships on behalf of any address (code).

5.b Minters for the Subscriptions contract

The SubscriptionsMinter and BMAIN DAO’s Agent App are Minters for the Subscriptions contract. (See Initial Deployment.)

The SubscriptionsMinter needs to have the minter role to call the mint (code) or claim (code) functions on the Subscriptions contract when someone wants to purchase Subscriptions (code) or claim Sponsorships (code), respectively.

BMAIN DAO’s Agent App needs to have the MinterRole to replace the SubscriptionsMinter contract if needed. This also means that BMAIN DAO’s Agent App can mint subscriptions on behalf of any address (code). It can also mark sponsorships from subscriptions as claimed (without actually minting them) for any address by calling Subscriptions.claim() directly.

5.c Replacing Minters

SponsorshipsMinter and SubscriptionsMinter can be replaced if issues are found after they have been deployed.

5.c.1 Detaching the SponsorshipsMinter contract

From the SponsorshipsMinter contract, disable purchases (code).

5.c.2 Detaching the SubscriptionsMinter contract

From the SubscriptionsMinter contract, disable purchases (code) and claims (code).

5.c.3 Adding New Minters

A new SponsorshipsMinter or SubscriptionsMinter can be added following the example of the deployment script.

6 Pausing Certain Functions

Pausing purchases, claims and context assignments can be useful in order to freeze balances so contracts can be replaced. See also replacing minters and upgrading the Sponsorships contract.

6.a Pauser Role

Because they inherit from NonTransferable, the creator of the Sponsorship (code) and Subscriptions (code, code) contracts acquires the Pauser role for the contracts (code, code) and can call the functions of the OpenZeppelin's Pausable module on them. See initial deployment.

6.b Pausing Functions in the Sponsorships contract

If the Sponsorships contract is paused, the mint (code) and assignContext (code) functions will be disabled (code, code).

6.c Pausing Functions in the Subscriptions contract

If the Subscriptions contract is paused, the mint (code) and claim (code) functions will be disabled (code, code, code).

7 Tokens Sent to Contracts by Mistake

The Sponsorships (code), Subscriptions (code), SponsorshipsMinter (code) and SubscriptionsMinter (code) contracts all inherit from FinanceManager which allows them transfer ERC20 tokens sent to them by mistake to the BMAIN DAO. (code, code, code, code).

8 Contract Ownership

Because the Sponsorships, Subscriptions, SponsorshipsMinter and SubscriptionsMinter contracts all inherit from FinanceManager, they also all inherit OpenZeppelin's Ownable (code). The initial owner will be the BMAIN DAO's Agent App.

8.a Functionality provided only to the Owner

- Collecting ERC20 tokens sent by mistake (code).
- Setting the token and price used for purchasing Sponsorships (code, code).

9 Initial Deployment

9.a Deployment Script

The script will be executed by BMAIN DAO's Agent app, making it the creator and owner of the Sponsorships (code), Subscriptions (code), SponsorshipsMinter (code), and SubscriptionsMinter (code) contracts.

9.b Constants

- Subscriptions are capped at 900,000 (code).
- The finance address for Sponsorships (code), Subscriptions (code), SponsorshipsMinter (code) and SubscriptionsMinter (code) contracts are set to BMAIN DAO's finance address (code).
- The initial token used for purchasing Sponsorships (code) and Subscriptions (code) is DAI (code).

9.c Minters

As the creator of the Sponsorships and Subscriptions contracts, BMAIN DAO's Agent App acquires the MinterRole for those contracts, which allows it to add minters to them.

The SponsorshipsMinter contract is added as a Minter for sponsorships (code). The SubscriptionsMinter contract is added as a Minter for both Subscriptions (code) and Sponsorships (code).

9.d Pausers

The creator of the Sponsorships and Subscriptions contracts (i.e. BMAIN DAO's Agent app) acquires the Pauser role and can call the functions of OpenZeppelin's Pausable module on those contracts.

10 Contracts/Libraries Used

10.a Sponsorships

- ./contracts/Finance.sol:Finance
  - Used by ./contracts/FinanceManager.sol:FinanceManager
- ./contracts/FinanceManager.sol:FinanceManager

    - Used by Sending Tokens to BMAIN DAO
- ./contracts/NonTransferable.sol:NonTransferable
    - Inherited by./contracts/Sponsorships.sol:Sponsorships
- ./contracts/Sponsorships.sol:Sponsorships
- ./contracts/openzeppelin-solidity/contracts/access/Roles.sol:Roles
    - Used by ./contracts/openzeppelin-solidity/contracts/access/roles/MinterRole.sol:MinterRole
    - Used by ./contracts/openzeppelin-solidity/contracts/access/roles/PauserRole.sol:PauserRole
- ./contracts/openzeppelin-solidity/contracts/access/roles/MinterRole.sol:MinterRole
    - Used by Minters for the Sponsorships contract
- ./contracts/openzeppelin-solidity/contracts/access/roles/PauserRole.sol:PauserRole
    - Used by Pausing Functions in the Sponsorships contract
- ./contracts/openzeppelin-solidity/contracts/lifecycle/Pausable.sol:Pausable
    - Used by Pausing Functions in the Sponsorships contract
- ./contracts/openzeppelin-solidity/contracts/math/SafeMath.sol:SafeMath
- ./contracts/openzeppelin-solidity/contracts/ownership/Ownable.sol:Ownable
    - Inherited by ./contracts/FinanceManager.sol:FinanceManager
- ./contracts/openzeppelin-solidity/contracts/token/ERC20/ERC20.sol:ERC20
    - Used by ./contracts/FinanceManager.sol:FinanceManager
- ./contracts/openzeppelin-solidity/contracts/token/ERC20/IERC20.sol:IERC20
    - Implemented by ./contracts/openzeppelin-solidity/contracts/token/ERC20/ERC20.sol:ERC20
- ./contracts/openzeppelin-solidity/contracts/utils/Address.sol:Address.isContract()
    - Used by ./contracts/FinanceManager.sol:FinanceManager

10.b Subscriptions

- ./contracts/Finance.sol:Finance
    - Used by ./contracts/FinanceManager.sol:FinanceManager
- ./contracts/FinanceManager.sol:FinanceManager
    - Used by Sending Tokens to BMAIN DAO
- ./contracts/NonTransferable.sol:NonTransferable
    - Inherited by ./contracts/NonTransferableCapped.sol:NonTransferableCapped
- ./contracts/NonTransferableCapped.sol:NonTransferableCapped
    - Inherited by ./contracts/Subscriptions.sol:Subscriptions
- ./contracts/Subscriptions.sol:Subscriptions
- ./contracts/openzeppelin-solidity/contracts/access/Roles.sol:Roles
    - Used by ./contracts/openzeppelin-solidity/contracts/access/roles/MinterRole.sol:MinterRole

    - Used by ./contracts/openzeppelin-solidity/contracts/access/roles/PauserRole.sol:PauserRole
- ./contracts/openzeppelin-solidity/contracts/access/roles/MinterRole.sol:MinterRole
    - Used by Minters for the Subscriptions contract
- ./contracts/openzeppelin-solidity/contracts/access/roles/PauserRole.sol:PauserRole
    - Used by Pausing Functions in the Subscriptions contract
- ./contracts/openzeppelin-solidity/contracts/lifecycle/Pausable.sol:Pausable
    - Used by Pausing Functions in the Subscriptions contract
- ./contracts/openzeppelin-solidity/contracts/math/SafeMath.sol:SafeMath
- ./contracts/openzeppelin-solidity/contracts/ownership/Ownable.sol:Ownable
    - Inherited by ./contracts/FinanceManager.sol:FinanceManager
- ./contracts/openzeppelin-solidity/contracts/token/ERC20/ERC20.sol:ERC20
    - Used by ./contracts/FinanceManager.sol:FinanceManager
- ./contracts/openzeppelin-solidity/contracts/token/ERC20/IERC20.sol:IERC20
    - Implemented by ./contracts/openzeppelin-solidity/contracts/token/ERC20/ERC20.sol:ERC20
- ./contracts/openzeppelin-solidity/contracts/utils/Address.sol:Address.isContract()
    - Used by ./contracts/FinanceManager.sol:FinanceManager

10.c SponsorshipsMinter

- ./contracts/Finance.sol:Finance
    - Used by ./contracts/FinanceManager.sol:FinanceManager
- ./contracts/FinanceManager.sol:FinanceManager
    - Used by Sending Tokens to BMAIN DAO
- ./contracts/NonTransferable.sol:NonTransferable
    - Inherited by./contracts/Sponsorships.sol:Sponsorships
- ./contracts/Sponsorships.sol:Sponsorships
    - Used by ./contracts/SponsorshipsMinter.sol:SponsorshipsMinter
- ./contracts/SponsorshipsMinter.sol:SponsorshipsMinter
- ./contracts/openzeppelin-solidity/contracts/access/Roles.sol:Roles
    - Used by ./contracts/openzeppelin-solidity/contracts/access/roles/MinterRole.sol:MinterRole
    - Used by ./contracts/openzeppelin-solidity/contracts/access/roles/PauserRole.sol:PauserRole
- ./contracts/openzeppelin-solidity/contracts/access/roles/MinterRole.sol:MinterRole
    - Used by Minters for the Sponsorships contract
- ./contracts/openzeppelin-solidity/contracts/access/roles/PauserRole.sol:PauserRole
    - Used by Pausing Functions in the Sponsorships contract
- ./contracts/openzeppelin-solidity/contracts/lifecycle/Pausable.sol:Pausable

  - Used by Pausing Functions in the Sponsorships contract
- ./contracts/openzeppelin-solidity/contracts/math/SafeMath.sol:SafeMath
- ./contracts/openzeppelin-solidity/contracts/ownership/Ownable.sol:Ownable
  - Inherited by ./contracts/FinanceManager.sol:FinanceManager
- ./contracts/openzeppelin-solidity/contracts/token/ERC20/ERC20.sol:ERC20
  - Used by ./contracts/FinanceManager.sol:FinanceManager
- ./contracts/openzeppelin-solidity/contracts/token/ERC20/IERC20.sol:IERC20
  - Implemented by ./contracts/openzeppelin-solidity/contracts/token/ERC20/ERC20.sol:ERC20
- ./contracts/openzeppelin-solidity/contracts/utils/Address.sol:Address.isContract()
  - Used by ./contracts/FinanceManager.sol:FinanceManager
  - Used by ./contracts/SponsorshipsMinter.sol:SponsorshipsMinter.setPurchaseToken()

10.d SubscriptionsMinter

- ./contracts/Finance.sol:Finance
  - Used by ./contracts/FinanceManager.sol:FinanceManager
- ./contracts/FinanceManager.sol:FinanceManager
  - Used by Sending Tokens to BMAIN DAO
- ./contracts/NonTransferable.sol:NonTransferable
  - Inherited by ./contracts/NonTransferableCapped.sol:NonTransferableCapped
  - Inherited by./contracts/Sponsorships.sol:Sponsorships
- ./contracts/NonTransferableCapped.sol:NonTransferableCapped
  - Inherited by ./contracts/Subscriptions.sol:Subscriptions
- ./contracts/Sponsorships.sol:Sponsorships
  - Used by ./contracts/SubscriptionsMinter.sol:SubscriptionsMinter
- ./contracts/Subscriptions.sol:Subscriptions
  - Used by ./contracts/SubscriptionsMinter.sol:SubscriptionsMinter
- ./contracts/SubscriptionsMinter.sol:SubscriptionsMinter
- ./contracts/openzeppelin-solidity/contracts/access/Roles.sol:Roles
  - Used by ./contracts/openzeppelin-solidity/contracts/access/roles/MinterRole.sol:MinterRole
  - Used by ./contracts/openzeppelin-solidity/contracts/access/roles/PauserRole.sol:PauserRole
- ./contracts/openzeppelin-solidity/contracts/access/roles/MinterRole.sol:MinterRole
  - Used by Minters for the Sponsorships contract
  - Used by Minters for the Subscriptions contract
- ./contracts/openzeppelin-solidity/contracts/access/roles/PauserRole.sol:PauserRole
  - Used by Pausing Functions in the Sponsorships contract
  - Used by Pausing Functions in the Subscriptions contract

- ./contracts/openzeppelin-solidity/contracts/lifecycle/Pausable.sol:Pausable
    - Used by Pausing Functions in the Sponsorships contract
    - Used by Pausing Functions in the Subscriptions contract
- ./contracts/openzeppelin-solidity/contracts/math/SafeMath.sol:SafeMath
- ./contracts/openzeppelin-solidity/contracts/ownership/Ownable.sol:Ownable
    - Inherited by ./contracts/FinanceManager.sol:FinanceManager
- ./contracts/openzeppelin-solidity/contracts/token/ERC20/ERC20.sol:ERC20
    - Used by ./contracts/FinanceManager.sol:FinanceManager
- ./contracts/openzeppelin-solidity/contracts/token/ERC20/IERC20.sol:IERC20
    - Implemented by ./contracts/openzeppelin-solidity/contracts/token/ERC20/ERC20.sol:ERC20
- ./contracts/openzeppelin-solidity/contracts/utils/Address.sol:Address.isContract()
    - Used by ./contracts/FinanceManager.sol:FinanceManager

11 Other Considerations

11.a Handling Upgrades and Exploits to the Sponsorships Contract

After pausing functions in the Sponsorships contract, a contract similar to OpenZeppelin's ERC20 Migrator could be used to migrate balances if the Sponsorships contract needed to be upgraded.

12 Github repo