Gnosis Safe Deep Dive

Deep Dives - Jan 24th

- Smart Contract Dev for Bulla Network 👨🏽‍💻
- Teach a Solidity Deep Dive Meetup 🤿
- Based in Denver 🏔️
- Github - Twitter - Streams

Repos:

- Flattened Contracts
- Foundry Test Tools

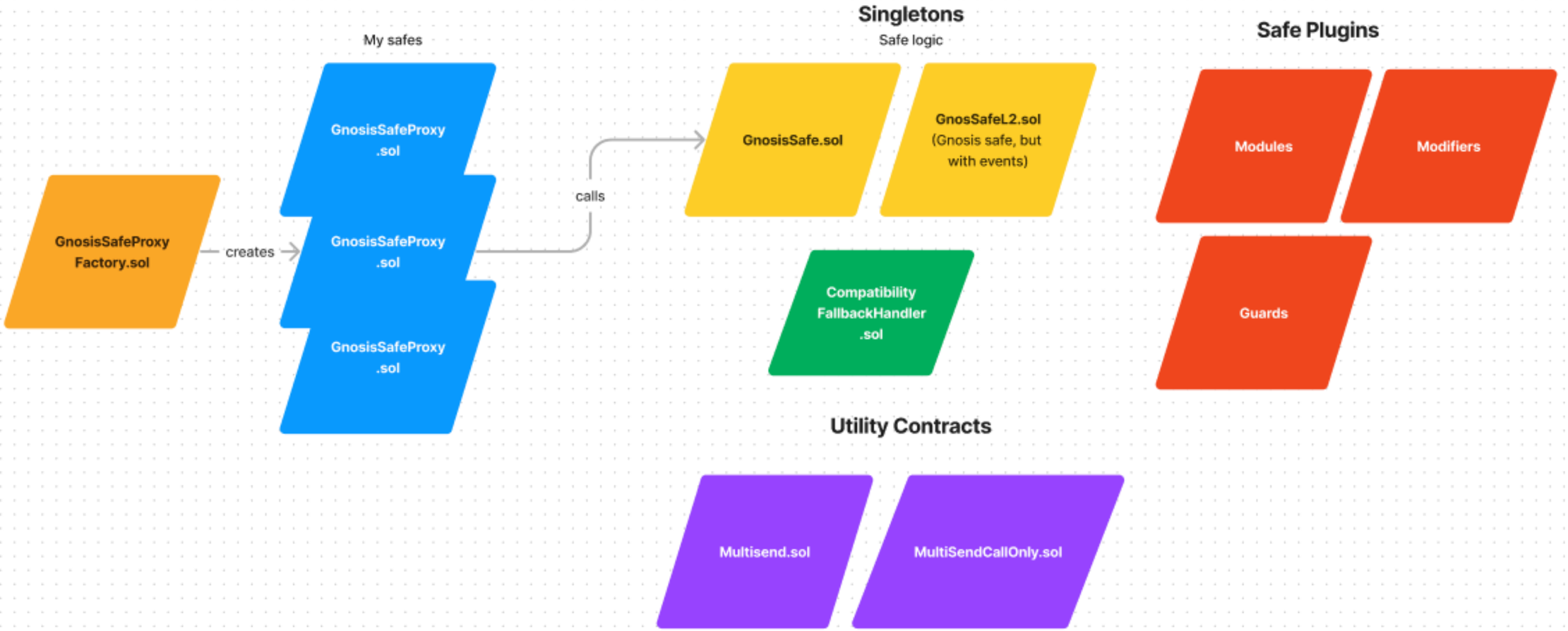
Agenda

- Safe Features
- The Safe ecosystem
- Contract deep dive

A Safe can…

- Execute any arbitrary transaction (either to other contracts or to itself)
- “Generate” valid EIP1271 signatures
- Validate multiple signatures (either EOA signatures (eth_sign, EIP712), EIP1271 signatures)
- Refund transaction executors
- Install ***Modules*** that bypass signature validation
- Install **Guards** that perform checks / interactions after every transaction

Safe Contract Ecosystem



Life Cycle

1. Setup

- Sets up owners and threshold
- Sets a fallback handler
- Can arbitrarily modify it's own storage by making a delegate call to the ***to*** param with the ***data*** param
- Refund the ***paymentReceiver*** ***payment*** (wei) of ***paymentToken***

```
function setup(
    address[] calldata _owners,
    uint256 _threshold,
    address to,
    bytes calldata data,
    address fallbackHandler,
    address paymentToken,
    uint256 payment,
    address payable paymentReceiver
) external {
    setupOwners(_owners, _threshold);
    if (fallbackHandler != address(0)) {
        internalSetFallbackHandler(fallbackHandler);
    }
    setupModules(to, data);

    if (payment > 0)
        handlePayment(payment, 0, 1, paymentToken, paymentReceiver);

    emit SafeSetup(msg.sender, _owners, _threshold, to, fallbackHandler);
}
```

Setup: Possibilities

- You could set up custom modules
- You could set up a transaction guard
- You could pre-approve n amount of transactions to be executed at any time
- You could spin-up a single-use DAO which starts a timer on initialization and can be self destructed after a certain amount of time
- …Execute any arbitrary transaction or callback on initialization 🤯
- (the world is your oyster)

NOTE: All these features would have to be written into a smart contract that would be deployed at a fixed address, then delegatecalled on initialization

Life Cycle

2. Execute transactions

Each transaction from execTransaction:

- Can either call or delegatecall
- Can have gas limits via the ***safeTxGas***, ***gasPrice***

```
function execTransaction(
    address to,
    uint256 value,
    bytes calldata data,
    Operation operation,
    uint256 safeTxGas,
    uint256 baseGas,
    uint256 gasPrice,
    address gasToken,
    address payable refundReceiver,
    bytes memory signatures
) public payable virtual returns (bool success);

function execTransactionFromModule(
    address to,
    uint256 value,
    bytes memory data,
    Operation operation
) public virtual returns (bool success);
```

Life Cycle

2.1 Tx encoding

- EOA: each tx is RLP encoded, hashed to a bytes32 value, then signed by a private key
- Safe TX’s are encoded to the EIP712 spec, -> keccak256 hashed -> then signed by EOAs or marked as signed by EOAs

EOA Tx

```
{
  "from": "0xEA674fdDe714fd979de3EdF0F56AA9716B898ec8",
  "to": "0xac03bb73b6a9e108530aff4df5077c2b3d481e5a",
  "gasLimit": "21000",
  "maxFeePerGas": "300",
  "maxPriorityFeePerGas": "10",
  "nonce": "0",
  "value": "10000000000",
  "chainId": "1"
}
```

Safe Tx

```
{
  "to": "0xac03bb73b6a9e108530aff4df5077c2b3d481e5a",
  "value": "10000000000",
  "data": "0x",
  "operation": "0",
  "baseGas": "21000",
  "gasPrice": "300",
  "gasToken": "10",
  "refundReceiver": "0",
  "nonce": "1",
  "chainId": "1",
  "safeAddress": "0x1230B3d59858296A31053C1b8562Ecf89A2f888b",
}
```

Life Cycle

2.2 Signature verification

- Signatures are encoded via the ***v*** value in the signature
- V == 0 means EIP1271 smart-contract signature
- V == 1 means approved hash (offline signing method)
- V > 30 means eth_sign
- Otherwise, it’s a sign_typed_data call

Dope signing algo! 💪

```
address lastOwner = address(0);
require(
    currentOwner > lastOwner &&
    owners[currentOwner] != address(0) &&
    currentOwner != SENTINEL_OWNERS,
    "GS026"
);
lastOwner = currentOwner;
```

If the signatures are from valid owners above the threshold, then the transaction is executed

Life Cycle

2.3 Guard Checks

- The safe delegates execution flow to a “***guard***”
- Checks happen before and after execution
- Guards are expected to revert if there is an issue.
- **Examples**

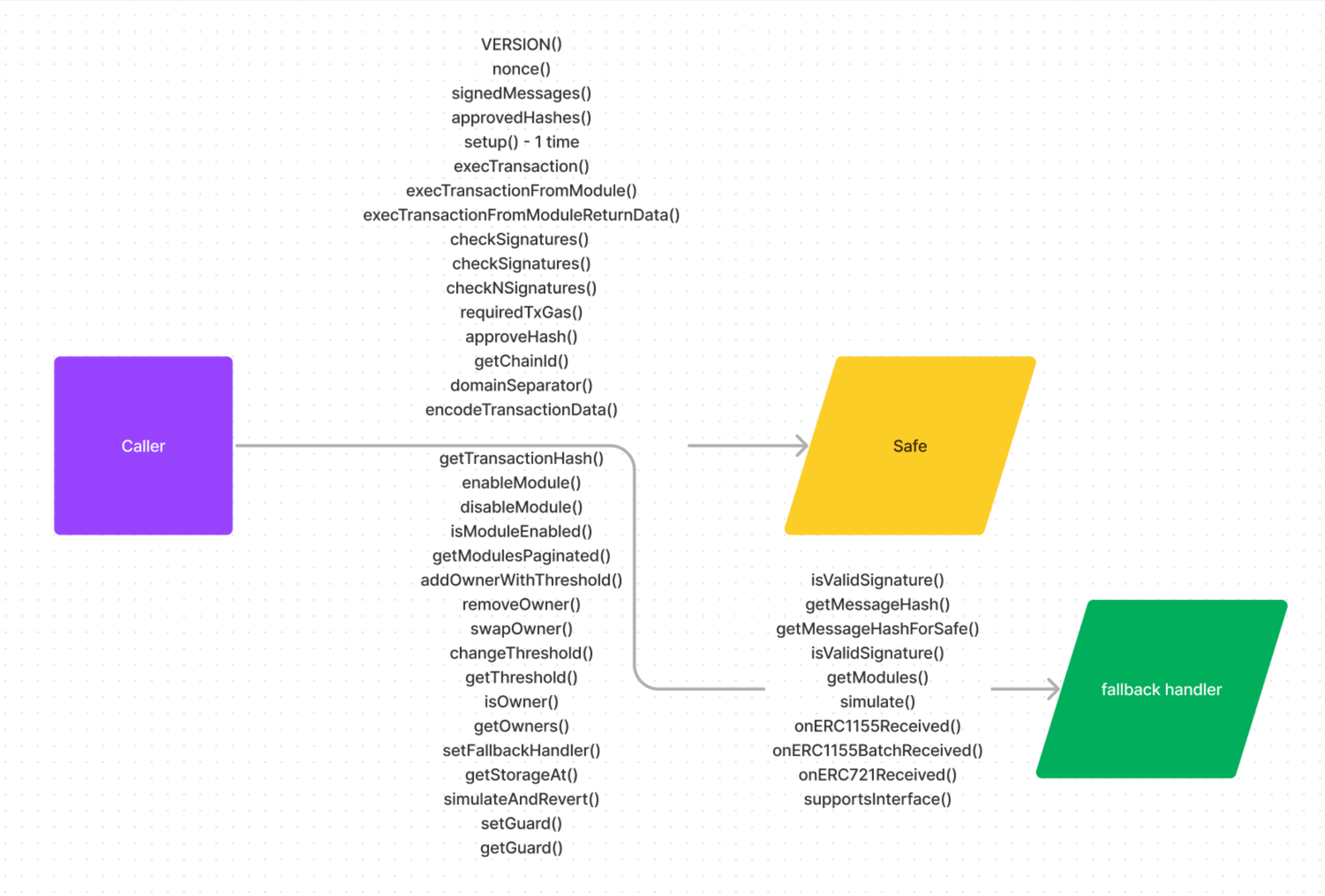
Life Cycle

2.4 Tx Execution

- Will simply execute a call to another contract - or a delegate call (which can modify it's own storage)
- Delegate calls can be dangerous, because they *can* modify the Safe's storage
- NOTE: Multisend transactions need to be delegate calls
- If gas > 0, that amount is refunded to the tx initiator

```
function execute(address to, uint256 value, bytes memory
 data, Operation operation, uint256 txGas)
    internal
    returns (bool success)
{
    if (operation == Operation.DelegateCall) {
        assembly {
            success := delegatecall(txGas, to, add(data, 0x20), mload(data), 0, 0)
        }
    } else {
        assembly {
            success := call(txGas, to, value, add(data, 0x20), mload(data), 0, 0)
        }
    }
}
```

Fallback Handler



Quick Hits:

- Signatures are gathered up off-chain, then submitted all at once via the Safe Transaction Service
- You can estimate the amount of gas required to execute a transaction by calling ***requiredTxGas***
- If the Safe Transaction Service ever goes down, you can manually sign transactions by calling ***getTransactionHash***, then calling **approveHash** with your signing wallet
- Call ***getMessageHashForSafe*** to encode a signed message for a Safe - this is compatible to Safe's version of eth_sign
- Each Safe deployed via the safe UI is a proxy contract - meaning its cost is minimal
- There is a module factory to deploy new modules (deploy gas cost is astronomically lower)

Ideas 🤔

- A contract can validate a Safe's EIP 1271 signature given a sorted array of owner signatures over the threshold (see CompatibilityFallbackHandler.isValidSignature)
- A custom fallback handler can be deployed to implement custom function calls and custom storage writes
- Install a trusted guard that performs an action on a Baal module on every transaction

Repos

- Flattened Contracts
- Foundry Test Tools