Ethereum Alarm Clock Documentation

Release 1.0.0

Piper Merriam

January 31, 2016

Contents

The Ethereum Alarm Clock is a service that allows scheduling of contract function calls at a specified block number in the future. These scheduled calls are executed by other nodes on the ethereum network in exchange for a small payment and reimbursment of gas costs.

The service is completely trustless, open source, and verifiable.

Contents:

CHAPTER 1

What is it?

Alarm is a service that operates on the Ethereum world computer.

Within the network there are two types of accounts.

- Contracts: Controlled by a piece of code that has been deployed to the Ethereum network.
- Private Key: Accounts that are controlled by a private key which is used to sign transaction.

The only difference between the capabilities of these two types of accounts is that only Private Key based accounts can initiate the execution of code which is done by sending a signed transaction into the Ethereum network.

A common requirement for software systems is execution of code at a specified time in the future. An example of one such mechanism is the crontab on unix based systems.

The Alarm service serves a similar purpose to Contracts on Ethereum. Since contracts are unable to initiate transactions themselves, the Alarm service sets up a marketplace that allows the operator of a private key based account to claim a reward in exchange for initiating the required transaction.

CHAPTER 2

Overview

The Ethereum Alarm service is a contract on the ethereum network that facilitates scheduling of function calls for a specified block in the future. It is designed to require zero trust between any of the users of the service, as well as providing no special access to any party (including the author of the service)

2.1 Scheduling Function Calls

Contracts, or individuals can schedule a function calls with the Alarm service by doing the following.

1. Schedule the function call with the service, providing basic information such as what function to call and when it should be called.
2. Register any call data that will be required to make the function call (optional for functions that have no arguments).

2.1.1 Call Scheduling

Function calls can be scheduled for any block at least 10 blocks *(~3 minutes)* in the future. Scheduling is done by calling the `scheduleCall` function on the scheduling contract. This function has a wide variety of call signatures that allow the scheduler to specify any of the following information.

1. Contract address the call should be executed on.
2. ABI signature of the function that should be called.
3. Bytes of call data that should be passed along.
4. Value in Ether to be sent with the call.
5. Target block number that the call should be executed on.
6. Number of blocks after the target block during which it still ok to execute the call. (between 64 - 255 blocks)
7. Required gas that must be provided with the executing transaction.
8. Stack depth check.
9. Payment amount in wei that will be paid to the executor of the call.
10. Donation amount in wei that will be paid to the creator of the Alarm service.

The scheduling transaction must also include enough ether to pay for the gas costs of the call as well as the payment and fee values.

2.2 Execution of scheduled calls

Scheduled function calls can be executed by anyone who wishes to initiate the transaction who inturn is paid whatever amount was specified as the payment value for the call.

2.2.1 Cost

In addition to the gas costs, schedulers are also encouraged include a payment amount for the executor of the call. This value can be specified by the scheduler, meaning that you may choose to offer any amount for the execution of your function.

The scheduling function uses the following defaults if specific values are not provided.

- Payment to the executor: **market value**
- Payment to the service creator: **1/100th market value**

The market value is determined by the market history of scheduled calls.

2.3 Guarantees

2.3.1 Will the call happen?

There are no guarantees that your function will be called. This is not a shortcoming of the service, but rather a fundamental limitation of the ethereum network. Nobody is capable of forcing a transaction to be included in a specific block.

The Alarm service has been designed such that it **should** become more reliable as more people use it.

However, it is entirely possible that certain calls will be missed due to unforseen circumstances. Providing a higher Payment amount is a potential way to get your scheduled call handled at a higher priority as it will be more profitable to execute.

2.3.2 Will I get paid for executing a call?

If you are diligent about how you go about executing scheduled calls then executing scheduled calls is guaranteed to be profitable. See the section on executing calls for more information.

CHAPTER 3

Scheduling

Call scheduling is the core of the Ethereum Alarm Service. Calls can be scheduled on any block at least 40 blocks *(~10 minutes)* in the future.

When a call is scheduled, the service deploys a new contract that represents the scheduled function call. This contract is referred to as the **call contract**. It holds all of the metadata associated with the call as well as the funds that will be used to pay for the call.

3.1 Lifecycle of a Call Contract

- **creation** - The call contract is created as part of call scheduling.
- **pending** - The call can be cancelled anytime prior to the claim stage.
- **claim** - The contract can be claimed which will grant the claimer exclusive rights to execution during the first 16 blocks of the call window.
- **frozen** - The contract is frozen, preventing cancellation starting 10 blocks before the call's target block through the last block in the call window.
- **execution** - The executing transaction is sent, which triggers the call contract to execute the function call.
- **payment** - payments are sent to the executor and the creator of the alarm service.
- **finalization** - The contract sends any remaining funds to the address which scheduled the call.

3.2 Scheduling the Call

Function calls are scheduled with the `scheduleCall` function on the Alarm service. This creates a new **call contract** that represents the function call. The primary signature for this function which accepts all allowed configuration options is as follows.

```
function scheduleCall(address contractAddress,
                      bytes4 abiSignature,
                      bytes callData,
                      uint16 requiredStackDepth,
                      uint8 gracePeriod,
                      uint[5] args) public returns (address);
```

The `uint[5] args` is an array of 5 `uint256` values which are respectively `callValue`, `targetBlock`, `requiredGas`, `basePayment`, `baseDonation`.

```
var (callValue, targetBlock, requiredGas, basePayment, baseDonation) = args;
```

There are a total of 10 available configuration options for a scheduled call.

- `address contractAddress`
- `bytes4 abiSignature`
- `bytes callData`
- `uint16 requiredStackDepth`
- `uint8 gracePeriod`
- `callValue`
- `targetBlock`
- `requiredGas`
- `basePayment`
- `baseDonation`

3.2.1 Call Configuration

address contractAddress:

The address of the contract that the call should be called on.

- **Default:** `msg.sender` of the scheduling transaction.

bytes4 abiSignature:

The 4 byte ABI signature of the function to be called.

- **Default:** N/A. If omitted this value is not used.

Note: This configuration is a convenience feature. It is perfectly fine to exclude this value and have the 4-byte function signature as part of the `callData`.

bytes callData:

- **Default:** N/A. If omitted this value is not used.

Note: If the abiSignature argument was specified then this should **not** include the 4-byte function signature. Otherwise the call will be **double** prefixed with the function signature which is likely not what you want.

uint callValue:

The amount in wei that will be sent as part of call execution.

- **Default:** 0

uint targetBlock:

The block number the call should be executed on. This must be at least 10 blocks in the future.

- **Default:** 10 blocks from the current block.

uint8 gracePeriod:

The number of blocks after `targetBlock` during which the call may still be executed. Cannot be less than 64 or greater than 255.

- **Default:** 255

uint requiredGas:

The amount of gas required to be sent along with the executing transaction. This value cannot be less than 200,000 or more than the block gas limit minus 200,000 (`block.gaslimit - 200000`).

Call execution requires that at least this amount of gas be provided

- **Default:** 200,000.

uint16 requiredStackDepth:

The number of call stack frames should be checked prior to execution of the function call cannot be less than 10 or greater than 1,000.

If the call is being executed by another contract, call execution will verify that the stack depth can be extended by this value.

- **Default:** 10

uint basePayment:

The base amount in wei that will be used to calculate the amount paid to the executor of the call.

- **Default:** The current *market value* of a scheduled call.

uint baseDonation:

The base amount in wei that will be used to calculate the amount donated to the creator of the service.

- **Default:** 1/100th of the current *market value* of a scheduled call.

3.2.2 Alternate Call Signatures

The `scheduleCall` function has many alternate call signatures that are intended for simpler use in common use cases.

- `scheduleCall()`

If called with no arguments then the scheduled call will execute a bare `addr.call()` where `addr` is the `msg.sender` from when the call was scheduled.

- `scheduleCall(bytes callData)`

In this case, the target of the call will be `msg.sender` from when the call was scheduled, but the `callData` will be passed into the call (`addr.call(callData)`)

- `scheduleCall(bytes4 abiSignature)`

This is very similar to `scheduleCall(bytes callData)` except that it can make it easy to execute a specific function that takes no arguments.

- `scheduleCall(uint256 callValue, address contractAddress)`
- `scheduleCall(address contractAddress, uint256 targetBlock, uint256 callValue)`

These signature can be used to easily schedule sending ether to another address.

The exhaustive list of signatures for `scheduleCall` can be found below.

3.2.3 Call Contract Address

Since each scheduled call is deployed as a standalone contract it can be useful to have the address for the call contract.

If the `scheduleCall` function is being used from within a contract, the address of the newly created call contract is returned. If instead, the function is being called directly in a transaction, the address of the call contract can be extracted from the transaction logs under the `CallScheduled` event.

Contract scheduling its own call

Contracts can take care of their own call scheduling.

```
contract Lottery {
    address scheduler; // set by some other mechanism.

    function beginLottery() public {
        ... // Do whatever setup needs to take place.

        // Now we schedule the picking of the winner.

        // the 4-byte signature of the local function we want to be called.
        bytes4 sig = bytes4(sha3("pickWinner()"));

        // approximately 24 hours from now
        uint targetBlock = block.number + 5760;

        // the 4-byte signature of the scheduleCall function.
        bytes4 scheduleCallSig = bytes4(sha3("scheduleCall(bytes4,uint256)"));

        scheduler.call(scheduleCallSig, sig, targetBlock)
    }

    function pickWinner() public {
        ...
    }
}
```

In this example `Lottery` contract, every time the `beginLottery` function is called, a call to the `pickWinner` function is scheduled for approximately 24 hours later (5760 blocks).

3.3 Upfront Payment

The service requires that you pay upfront for all costs associated with call scheduling. This value is referred to as the **endowment**. Without intimate knowledge of how all of these things are calculated it can be difficult to determine how much to send.

One nice part about the service is that you can just send extra and anything unused will be returned to you. This is generally a good strategy since you are at no risk of losing your ether and it prevents situations where you come in slightly under the required endowment and have your call rejected.

The following functions are available to assist in computing this ether value.

- `getMinimumEndowment() constant returns (uint)`
- `getMinimumEndowment(uint basePayment) constant returns (uint)`

- `getMinimumEndowment(uint basePayment, uint baseDonation) constant returns (uint)`
- `getMinimumEndowment(uint basePayment, uint baseDonation, uint callValue) constant returns (uint)`
- `getMinimumEndowment(uint basePayment, uint baseDonation, uint callValue, uint requiredGas) constant returns (uint)`

3.4 Call Data

If a function call requires arguments then you have two options available.

- Provide the `bytes` as the `callData` argument at the time of scheduling.
- Register the `bytes` after the call has already been scheduled.

The call contract allows for call data registration via two mechanisms. The primary mechanism is through the fallback function on the contract. This will set the call data as the full call data of the transaction.

For example, if you were registering the call data for a function with the signature `myFunction(uint count, bytes32 reason)` you could do it with the following solidity code.

```
address scheduler = 0x...;
// schedule the call
address callContract = scheduler.call(...);

// Register the call data
scheduler.call(12345, 'abcde');
```

Alternatively you can use the `registerData()` function which will strip the first four bytes off of `msg.data` and use the remainder as the call data.

In solidity, this would look something like the following.

```
contract Example {
    function doDataRegistration() public {
        uint arg1 = 3;
        int arg2 = -1;
        to.call(bytes4(sha3("registerData()")), arg1, arg2);
    }
}
```

Once data has been registered, it cannot be modified. Attempts to do so will result in an exception.

Note: The `call()` function on an address in solidity does not do any ABI encoding, so in cases where a scheduled call must pass something like a `bytes` variable, you will need to handle the ABI encoding yourself.

3.5 Cancelling a call

A scheduled call can be cancelled by its scheduler either before the claim window begins.

- **Solidity Function Signature:** `cancel()`

This will cause the call to be set as **cancelled**, which will return any funds currently being held by the contract.

A call may also be cancelled after the call window if it has not been executed.

3.6 Looking up a Call

You can lookup whether a particular address is a known scheduled call with the `isKnownCall` function.

- **Solidity Function Signature:** `isKnownCall(address callAddress) returns (bool)`

Returns a boolean as to whether this address represents a known scheduled call.

3.7 Helper Functions

The following getters can be used to return the constant values that are used by the service programatically.

- `callAPIVersion() constant returns (uint)`

Returns the version of the Alarm service.

- `getMinimumGracePeriod() constant returns (uint)`

The smallest value allowed for the `gracePeriod` of a scheduled call.

- `getDefaultDonation() constant returns (uint)`

The default payment value for scheduled calls.

- `getMinimumCallGas() constant returns (uint)`

The minimum allowed value for `requiredGas`

- `getMaximumCallGas() constant returns (uint)`

The maximum allowed value for `requiredGas`. This value is computed as `block.gaslimit - getMinimumCallGas()`

- `getDefaultRequiredGas() constant returns (uint)`

The default value for `requiredGas`

- `isKnownCall(address callAddress) constant returns (bool)`

Returns whether this address was a call contract that was deployed by the alarm service. This can be useful if you need to use the service to interact with priviledged functions as you can verify that the address that is calling you is in fact a legitimate call contract.

- `getFirstSchedulableBlock() constant returns (uint)`

Returns the earliest block number in the future on which a call may be scheduled.

- `getMinimumStackCheck() constant returns (uint16)`

The minimum allowed value for `requiredStackDepth`.

- `getMaximumStackCheck() constant returns (uint16)`

The maximum allowed value for `requiredStackDepth`.

- `getDefaultStackCheck() constant returns (uint16)`

The default value for the `requiredStackDepth` of a scheduled call.

- `getDefaultGracePeriod() constant returns (uint8)`

The default value for the `gracePeriod` of a scheduled call.

CHAPTER 4

Call Contract API

4.1 Properties of a Call Contract

A call contract for a scheduled call has the following publicly accessible values.

- **address contractAddress:** the address of the contract the function should be called on.
- **address schedulerAddress:** the address who scheduled the call.
- **uint targetBlock:** the block that the function should be called on.
- **uint8 gracePeriod:** the number of blocks after the `targetBlock` during which it is stll ok to execute the call.
- **uint anchorGasPrice:** the gas price that was used when the call was scheduled.
- **uint requiredGas:** the amount of gas that must be sent with the executing transaction.
- **uint basePayment:** the amount in wei that will be paid to the address that executes the function call.
- **uint baseFee:** the amount in wei that will be paid the creator of the Alarm service.
- **bytes4 abiSignature:** the 4 byte ABI function signature of the function on the `contractAddress` for this call.
- **bytes callData:** the data that will be passed to the called function.
- **bytes callValue:** the value in wei that will be sent with this call
- **uint16 requiredStackDepth:** the depth by which the stack must be increasable at the time of execution.
- **bool wasCalled:** whether the call was called.
- **bool wasSuccessful:** whether the call was successful during execution.
- **bool isCancelled:** whether the call was cancelled.
- **address claimer:** the address that has claimed this contract.
- **uint claimAmount:** the amount that the claimer agreed to execute the contract for.
- **uint claimerDeposit:** the amount that the claimer has put up for deposit.

4.1.1 Contract Address

address contractAddress

The address of the contract that the scheduled function call should be executed on. Retrieved with the `contractAddress` function.

- **Solidity Function Signature:** `contractAddress() returns (address)`

4.1.2 Scheduler Address

address schedulerAddress

The address that the scheduled function call. Retrieved with the `schedulerAddress` function.

- **Solidity Function Signature:** `schedulerAddress() returns (address)`

4.1.3 Target Block

uint targetBlock

The block number that this call should be executed on. Retrieved with the `targetBlock` function.

- **Solidity Function Signature:** `targetBlock() returns (uint)`

4.1.4 Grace Period

uint8 gracePeriod

The number of blocks after the **targetBlock** that it is still ok to execute this call. Retrieved with the `gracePeriod` function.

- **Solidity Function Signature:** `gracePeriod() returns (uint8)`

4.1.5 Anchor Gas Price

uint anchorGasPrice

The value of `tx.gasprice` that was used to schedule this function call. Retrieved with the `anchorGasPrice` function.

- **Solidity Function Signature:** `anchorGasPrice() returns (uint)`

4.1.6 Required Gas

uint requiredGas

The amount of gas that must be sent with the executing transaction. Retrieved with the `requiredGas` function.

- **Solidity Function Signature:** `requiredGas() returns (uint)`

4.1.7 Base Payment

uint basePayment

The base amount, in wei that the call executor's payment will be calculated from. Retrieved with the `basePayment` function.

- **Solidity Function Signature:** `basePayment() returns (uint)`

4.1.8 Base Fee

uint baseFee

The base amount, in wei that the fee to the creator of the alarm service will be calculate from. Retrieved with the `baseFee` function.

- **Solidity Function Signature:** `baseFee() returns (uint)`

4.1.9 ABI Signature

bytes4 abiSignature

The ABI function signature that should be used to execute this function call. Retrieved with the `abiSignature` function.

- **Solidity Function Signature:** `abiSignature() returns (uint)`

4.1.10 Call Data

bytes callData

The full call data that will be used for this function call. Retrieved with the `callData` function.

- **Solidity Function Signature:** `callData() returns (bytes)`

4.1.11 Call Value

uint callValue

The amount in wei that will be sent with the function call. Retrieved with the `callValue` function.

- **Solidity Function Signature:** `callValue() returns (bytes)`

4.1.12 Was Called

bool wasCalled

Boolean as to whether this call has been executed. Retrieved with the `wasCalled` function.

- **Solidity Function Signature:** `wasCalled() returns (bool)`

4.1.13 Was Successful

bool wasSuccessful

Boolean as to whether this call was successful. This indicates whether the called contract returned without error. Retrieved with the `wasSuccessful` function.

- **Solidity Function Signature:** `wasSuccessful() returns (bool)`

4.1.14 Is Cancelled

bool isCancelled

Boolean as to whether this call has been cancelled. Retrieved with the `isCancelled` function.

- **Solidity Function Signature:** `isCancelled() returns (bool)`

4.1.15 Claimer

address claimer

Address of the account that has claimed this call for execution. Retrieved with the `claimer` function.

- **Solidity Function Signature:** `claimer() returns (address)`

4.1.16 Claim Amount

uint claimAmount

Ammount that the `claimer` has agreed to pay for the call. Retrieved with the with the `claimAmount` function.

- **Solidity Function Signature:** `claimAmount() returns (uint)`

4.1.17 Claim Deposit

uint claimerDeposit

Ammount that the `claimer` put down as a deposit. Retrieved with the with the `claimerDeposit` function.

- **Solidity Function Signature:** `claimerDeposit() returns (uint)`

4.2 Functions of a Call Contract

4.2.1 Cancel

Cancels the scheduled call, suiciding the call contract and sending any funds to the scheduler's address. This function cannot be called from 265 blocks prior to the **target block** for the call through the end of the grace period.

Before the call, only the scheduler may cancel the call. Afterwards, anyone may cancel it.

- **Solidity Function Signature:** `cancel() public`

4.2.2 Execute

Triggers the execution of the call. This can only be done during the window between the `targetBlock` through the end of the `gracePeriod`. If the call has been claimed, then only the claiming address can execute the call during the first 16 blocks. If the claming address does not execute the call during this time, anyone who subsequently executes the call will receive their deposit.

- **Solidity Function Signature:** `execute() public`

4.2.3 Claim Helpers

- `function firstClaimBlock() constant returns (uint)`

The first block during which the call may be claimed.

- `function maxClaimBlock() constant returns (uint)`

The block during the claim window when the call will be worth the full `basePayment` value.

- `function lastClaimBlock() constant returns (uint)`

The last block during which the call may be claimed.

- `function getClaimAmountForBlock() constant returns (uint)`
- `function getClaimAmountForBlock(uint block_number) constant returns (uint)`

Returns the paymend value for a block in the claim window. If called with no argument it uses the current block.

CHAPTER 5

Call Execution

Call execution is the process through which scheduled calls are executed at their desired block number. After a call has been scheduled, it can be executed by account which chooses to initiate the transaction. In exchange for executing the scheduled call, they are paid a small fee of approximately 1% of the gas cost used for executing the transaction.

5.1 Executing a call

Use the `execute` function to execute a scheduled call. This function is present on the call contract itself (as opposed to the scheduling service).

- **Solidity Function Signature:** `execute() public`

When this function is called, the following things happen.

1. A few checks are done to be sure that all of the necessary pre-conditions pass. If any fail, the function exits early without executing the scheduled call:
    - the call has not already been called.
    - the call has not been cancelled.
    - the transaction has at least `requiredGas` in gas.
    - the stack depth can be extended sufficiently deep for `requiredStackDepth`
    - the current block number is within the range this call is allowed to be executed.
    - the caller is allowed to execute the function (see claiming)
2. The call is executed
3. The gas cost and fees are computed and paid.
4. The call contract sends any remaining funds to the scheduling address.

5.1.1 Payment

Each scheduled call sets its own payment value. This can be looked up with the `basePayment` accessor function.

The final payment value for executing the scheduled call is the `basePayment` multiplied by a scalar value based on the difference between the gas price of the executing transaction and the gas price that was used to schedule the transaction. The formula for this scalar is such that the lower the gas price of the executing transaction, the higher the payment.

5.1.2 Setting transaction gas and gas price

Each call contract has a `requiredGas` property. Execution of a call requires at least this amount of gas be sent with the transaction.

This gas value should be used in conjuction with the `basePayment` and `baseFee` amounts with respect to the ether balance of the call contract. The provided gas for the transaction should not exceed `(balance - 2 * (basePayment + baseFee)) / gasPrice` if you want to guarantee that you will be fully reimbursed for gas expenditures.

5.1.3 Getting your payment

Payment for executing a call is sent to you as part of the executing transaction, as well as being logged by the `CallExecuted` event.

5.2 Determining what scheduled calls are next

You can query the Alarm service for the call key of the next scheduled call on or after a specified block number using the `getNextCall` function

- **Solidity Function Signature:** `getNextCall(uint blockNumber) returns (address)`

Since there may be multiple calls on the same block, it is best to also check if the call has any *siblings* using the `getNextCallSibling` function. This function takes a call contract address and returns the address that is scheduled to come next.

When checking for additional calls in this manner, you should check the target block of each subsequent call to be sure it is within a range that you care about.

- **Solidity Function Signature:** `getNextCallSibling(address callAddress) returns (address)`

Note: 10 blocks into the future is a good range to monitor since new calls must always be scheduled at least 10 blocks in the future.

5.3 The Freeze Window

The 10 blocks prior to a call's target block are called the **freeze window**. During this window, nothing about a call can change. This means that it cannot be cancelled or claimed.

5.4 Claiming a call

Claiming a call is the process through which you as a call executor can guarantee the exclusive right to execute the call during the first 16 blocks of the call window for the scheduled call. As part of the claim, you will need to put down a deposit, which is returned to you if you when you execute the call. Failing to execute the call will forfeit your deposit.

5.4.1 Claim Amount

A call can be claimed during the 255 blocks prior to the freeze window. This period is referred to as the claim window. The amount that you are agreeing to be paid for the call is based on whichever block the call is claimed on. The amount can be calculated using the following formula.

- Let `i` be the index of the block within the 255 block claim window.
- Let `basePayment` be the payment amount specified by the call contract.
- If within the first 240 blocks of the window: `payment = basePayment * i / 240`
- If within the last 15 blocks of the window: `payment = basePayment`

This formula results in a linear growth from 0 to the full `basePayment` amount over the course of the first 240 blocks in the claim window. The last 15 blocks are all set at the full `basePayment` amount.

A claim must be accompainied by a deposit that is at least twice the call's `basePayment` amount.

5.4.2 Getting your Deposit Back

If you claim a call and do not execute it within the first 16 blocks of the call window, then you will risk losing your deposit. Once the first 16 blocks have passed, the call can be executed by anyone. At this point, the first person to execute the call will receive the deposit as part of their payment (and incentive to pick up claimed calls that have not been called).

5.4.3 Claim API

To claim a contract

- **Solidity Function Signature:** `claim()`

To check what the `claimAmount` will be for a given block number use the `getClaimAmountForBlock` function. This will return an amount in wei that represents the base payment value for the call if claimed on that block.

- **Solidity Function Signature:** `getClaimAmountForBlock(uint blockNumber)`

This function also has a shortcut that uses the current block number

- **Solidity Function Signature:** `getClaimAmountForBlock()`

You can check if a call has already been claimed with the `claimer` function. This function will return either the empty address `0x0` if the call has not been claimed, or the address of the claimer if it has.

- **Solidity Function Signature:** `claimer() returns (address)`

5.5 Safeguards

There are a limited set of safeguards that Alarm protects those executing calls from.

- Ensures that the call cannot cause the executing transaction to fail due to running out of gas (like an infinite loop).
- Ensures that the funds to be used for payment are locked during the call execution.

5.6 Tips for executing scheduled calls

The following tips may be useful if you wish to execute calls.

5.6.1 Look in the next 265 blocks

Calls within this window are likely claimable.

5.6.2 Calls are frozen during the 10 blocks prior to the target block

Once a call enters the freeze window it is immutable until call execution.

5.6.3 No cancellation in next 265 blocks

Since calls cannot be cancelled less than 265 blocks in the future, you don't need to check cancellation status during the 265 blocks prior to its target block.

5.6.4 Check that it was not already called

If you are executing a call after the target block but before the grace period has run out, it is good to check that it has not already been called.

5.6.5 Compute how much gas to provide

If you want to guarantee that you will be 100% reimbursed for your gas expenditures, then you need to compute how much gas the contract can pay for. The *overhead* involved in execution is approximately 140,000 gas. The following formula should be a close approximation to how much gas a contract can afford.

- let `gasPrice` be the gas price for the executing transaction.
- let `balance` be the ether balance of the contract.
- let `claimerDeposit` be the claimer's deposit amount.
- let `basePayment` be the base payment amount for the contract. This may either be the value specified by the scheduler, or the `claimAmount` if the contract has been claimed.
- `gas = (balance - 2 * basePayment - claimerDeposit) / gasPrice`

Reliability

One of the ways to assess the reliability of the Alarm service is to check the Reliability Canary. This is a contract which continually reschedules a call to itself every 480 blocks (approximately 2 hours). If any of these function calls is ever missed the canary *dies*.

http://canary.ethereum-alarm-clock.com/

A dead canary means that at some point one of the calls that the canary scheduled was not executed.

CHAPTER 7

Execution Client

Warning: This tool should be considered alpha software and comes with all of the caveats and disclaimers that one might expect with early stage software. Use at your own risk.

The easiest way to start executing calls is to use the Ethereum Alarm Clock Client.

Installation can be done with pip

```
$ pip install ethereum-alarm-clock-client
```

Once the package is installed, a new command line tool `eth_alarm` should be available.

7.1 Running the Scheduler

The execution scheduler is a process that monitors the alarm service for upcoming scheduled function calls and executes them at their target block.

The scheduler requires an *unlocked* ethereum client with the JSON-RPC server enabled to be running on localhost.

```
$ eth_alarm scheduler
BLOCKSAGE: INFO: 2015-12-23 15:31:26,920 > Starting block sage
BLOCKSAGE: INFO: 2015-12-23 15:31:38,143 > Heartbeat: block #328 : block_time: 1.90237202068
BLOCKSAGE: INFO: 2015-12-23 15:31:43,623 > Heartbeat: block #335 : block_time: 1.75782920308
SCHEDULER: INFO: 2015-12-23 15:31:56,415 Tracking call: 0xa4a1b0d99e5271dd236a7f2abe30f81bba67dd90
CALL-0XA4A1B0D99E5271DD236A7F2ABE30F81BBA67DD90: INFO: 2015-12-23 15:31:56,415 Sleeping until 377
BLOCKSAGE: INFO: 2015-12-23 15:31:58,326 > Heartbeat: block #340 : block_time: 1.89721174014
BLOCKSAGE: INFO: 2015-12-23 15:32:06,473 > Heartbeat: block #346 : block_time: 2.07706735856
BLOCKSAGE: INFO: 2015-12-23 15:32:12,427 > Heartbeat: block #352 : block_time: 1.78518210439
BLOCKSAGE: INFO: 2015-12-23 15:32:24,904 > Heartbeat: block #357 : block_time: 1.67715797869
BLOCKSAGE: INFO: 2015-12-23 15:32:32,134 > Heartbeat: block #363 : block_time: 2.02664816647
BLOCKSAGE: INFO: 2015-12-23 15:32:41,400 > Heartbeat: block #368 : block_time: 1.70622547582
BLOCKSAGE: INFO: 2015-12-23 15:32:48,291 > Heartbeat: block #373 : block_time: 1.59583837187
BLOCKSAGE: INFO: 2015-12-23 15:32:53,134 > Heartbeat: block #378 : block_time: 1.51536617309
CALL-0XA4A1B0D99E5271DD236A7F2ABE30F81BBA67DD90: INFO: 2015-12-23 15:32:55,419 Entering call loop
CALL-0XA4A1B0D99E5271DD236A7F2ABE30F81BBA67DD90: INFO: 2015-12-23 15:32:55,452 Attempting to execute
CALL-0XA4A1B0D99E5271DD236A7F2ABE30F81BBA67DD90: INFO: 2015-12-23 15:32:59,473 Transaction accepted.
```

- The process will log a *heartbeat* every 4 blocks (1 minute).
- When an upcoming call is found that is unclaimed, the scheduler will claim the call. (more on this later)
- When an upcoming scheduled call is found within the next 40 blocks it will print a notice that it is now tracking that call.

- When the target block is imminent (2 blocks) a notice that it is entering the *call loop* is logged.

7.1.1 Call Claiming

The scheduling client will claim scheduled calls. The logic for claiming is roughly the following.

- Let `N` be a number between 0 and 255 that represents the current location in the call's claim window.
- If the claim value at `N` is not at least enough to pay for the claiming transaction and `N` is less than 240 the call is ignored.
- If the call is profitable at `N` or `N` is greater than 240, a random number `x` is generated between 0 and 255.
- If `x` is less than `N` then the client will attempt to claim the call. Otherwise the call is ignored.

7.1.2 Call Execution

Once the call enters the call window the client will check whether the call is claimed.

- If the call is claimed and the current coinbase is the claiming address then the client will attempt to execute the call.
- If the call is claimed and the claiming address is not the current coinbase the client will wait until the call enters free-for-all mode.
- If the call is unclaimed then the client will attempt to execute it.

Once the client has sent a transaction attempting to execute the call it will wait for the transaction receipt and no further attempts to execute the call will be made.

7.1.3 Checks

The client implements all of the following

- Don't claim cancelled calls
- Don't execute cancelled calls
- Don't execute calls that have already been executed.
- Don't execute calls that don't have enough ether to pay for the transaction.

7.1.4 Other Things You Should Know

- The script only logs that the transaction was accepted which is not the same as successfully executing the call. see issue 1

7.2 Running a server

The scheduler runs nicely on the smallest AWS EC2 instance size. The following steps should get an EC2 instance provisioned with the scheduler running.

7.2.1 1. Setup an EC2 Instance

- Setup an EC2 instance running Ubuntu. The smallest instance size works fine.
- Add an extra volume to store your blockchain data. 16GB should be sufficient for the near term future.
- Optionally mark this volume to persist past termination of the instance so that you can reuse your blockchain data.
- Make sure that the security policy leaves *30303* open to connections from the outside world.

7.2.2 2. Provision the Server

- `sudo apt-get update --fix-missing`
- `sudo apt-get install -y supervisor`
- `sudo apt-get install -y python-dev python build-essential libreadline-gplv2-dev libncursesw5-dev libssl-dev libsqlite3-dev tk-dev libgdbm-dev libc6-dev libbz2-dev python-virtualenv`

7.2.3 3. Mount the extra volume

The following comes from the AWS Documentation and will only work verbatim if your additional volume is `/dev/xvdb`.

- `sudo mkfs -t ext4 /dev/xvdb`
- `sudo mkdir -p /data`
- `sudo mount /dev/xvdb /data`
- `sudo mkdir -p /data/ethereum`
- `sudo chown ubuntu /data/ethereum`

Modify */etc/fstab* to look like the following. This ensures the extra volume will persist through restarts.

```
#/etc/fstab
LABEL=cloudimg-rootfs   /        ext4   defaults,discard        0 0
/dev/xvdb       /data   ext4    defaults,nofail        0        2
```

Run `sudo mount -a` If you don't get any errors then you haven't borked your `etc/fstab`

7.2.4 4. Install Geth

Install the go-ethereum client.

- `sudo apt-get install -y software-properties-common`
- `sudo add-apt-repository -y ppa:ethereum/ethereum`
- `sudo apt-get update`
- `sudo apt-get install -y ethereum`

7.2.5 5. Install the Alarm Client

Install the Alarm client.

- `mkdir -p ~/alarm-0.6.0`
- `cd ~/alarm-0.6.0`
- `virtualenv env && source env/bin/activate`
- `pip install ethereum-alarm-clock-client`

7.2.6 6. Configure Supervisord

Supervisord will be used to manage both `geth` and `eth_alarm`.

Put the following in `/etc/supervisord/conf.d/geth.conf`

```
[program:geth]
command=geth --datadir /data/ethereum --unlock 0 --password /home/ubuntu/geth_password --rpc --fast
user=ubuntu
stdout_logfile=/var/log/supervisor/geth-stdout.log
stderr_logfile=/var/log/supervisor/geth-stderr.log
```

Put the following in `/etc/supervisord/conf.d/scheduler-v6.conf`

```
[program:scheduler-v6]
user=ubuntu
command=/home/ubuntu/alarm-0.6.0/env/bin/eth_alarm scheduler --client rpc --address 0xe109ecb193841a
directory=/home/ubuntu/alarm-0.6.0/
environment=PATH="/home/ubuntu/alarm-0.6.0/env/bin"
stdout_logfile=/var/log/supervisor/scheduler-v6-stdout.log
stderr_logfile=/var/log/supervisor/scheduler-v6-stderr.log
autorestart=true
autostart=false
```

7.2.7 7. Generate geth account

Use the following command to generate an account. The `--datadir` argument is important, otherwise the generated account won't be found by our geth process being run by supervisord.

- `$ geth --datadir /data/ethereum account new`

Place the password for that account in `/home/ubuntu/geth_password`.

You will also need to send this account a few ether. Twice the maximum transaction cost should be sufficient.

7.2.8 8. Turn it on

Reload supervisord so that it finds the two new config files.

- `sudo supervisord reload`

You'll want to wait for `geth` to fully sync with the network before you start the `scheduler-v6` process.

7.2.9 9. Monitoring

You can monitor these two processes with `tail`

- `tail -f /var/log/supervisor/geth*.log`
- `tail -f /var/log/supervisor/scheduler-v6*.log`

CHAPTER 8

Costs

The Alarm service operates under a **scheduler pays** model, which means that the scheduler of a call is responsible for paying up front for all costs associated with execution. These funds are held within the call contract until execution or cancellation.

8.1 Call Payment and Donation

When a call is scheduled, the scheduler can either provide values for the payment and donation, or leave them off in favor of using the default values.

The account which executes the scheduled call is reimbursed 100% of the gas cost + payment for their service.

The donation is sent to the creator of the Alarm service.

8.2 Default Payment Value

The default used for the payment value for each scheduled call is ultimately decided by the open market.

- If calls are being claimed early in the claim window the value will decrease.
- If calls are being claimed late in the claim window the value will increase

Note: This algorithm is a work in progress. If you have ideas on ways to improve this feel free to reach out to me.

8.3 Gas Costs

- Scheduling a function call takes approximately 1.1 million gas.
- Execution adds approximately 100,000 gas of overhead.

8.4 The Gas Price Scalar multiplier

Both the payment and the donation are multiplied by the **GasPriceScalar**.

GasPriceScalar is a multiplier that ranges from 0 - 2 which is based on the difference between the gas priced used for call execution and the gas price used during call scheduling. This number incentivises the call executor to use as low a gas price as possible.

This multiplier is computed with the following formula.

- *IF* `gasPrice > anchorGasPrice`

`anchorGasPrice / gasPrice`

- *IF* `gasPrice <= anchorGasPrice`

`anchorGasPrice / (2 * anchorGasPrice - gasPrice)`

Where:

- **anchorGasPrice** is the `tx.gasprice` used when the call was scheduled.
- **gasPrice** is the `tx.gasprice` used to execute the call.

At the time of call execution, the `anchorGasPrice` has already been set, so the only value that is variable is the `gasPrice` which is set by the account executing the transaction. Since the scheduler is the one who ends up paying for the actual gas cost, this multiplier is designed to incentivize the caller using the lowest gas price that can be expected to be reliably picked up and promptly executed by miners.

Here are the values this formula produces for a `baseGasPrice` of 20 and a `gasPrice` ranging from 10 - 40;

gasPrice	multiplier
15	1.20
16	1.17
17	1.13
18	1.09
19	1.05
20	1.00
21	0.95
22	0.91
23	0.87
24	0.83
25	0.80
26	0.77
27	0.74
28	0.71
29	0.69
30	0.67
31	0.65
32	0.63
33	0.61
34	0.59
35	0.57
36	0.56
37	0.54
38	0.53
39	0.51
40	0.50

You can see from this table that as the `gasPrice` for the executing transaction increases, the total payout for executing the call decreases. This provides a strong incentive for the entity executing the transaction to use a reasonably low value.

Alternatively, if the `gasPrice` is set too low (potentially attempting to maximize payout) and the call is not picked up by miners in a reasonable amount of time, then the entity executing the call will not get paid at all. This provides a strong incentive to provide a value high enough to ensure the transaction will be executed.

Contract ABI

Beyond the simplest use cases, the use of `address.call` to interact with the Alarm service is limiting. Beyond the readability issues, it is not possible to get the return values from function calls when using `call()`.

By using an abstract solidity contract which defines all of the function signatures, you can easily call any of the Alarm service's functions, letting the compiler handle computation of the function ABI signatures.

9.1 Abstract Solidity Contracts

The following abstract contracts can be used alongside your contract code to interact with the Alarm service.

9.1.1 Abstract Scheduler Contract Source Code

The following abstract solidity contract can be used to interact with the scheduling contract from a solidity contract.

```
contract SchedulerAPI {
    /*
     *  Call Scheduling API
     */
    function getMinimumGracePeriod() constant returns (uint);
    function getDefaultDonation() constant returns (uint);
    function getMinimumCallGas() constant returns (uint);
    function getMaximumCallGas() constant returns (uint);
    function getMinimumEndowment() constant returns (uint);
    function getMinimumEndowment(uint basePayment) constant returns (uint);
    function getMinimumEndowment(uint basePayment, uint baseDonation) constant returns (uint);
    function getMinimumEndowment(uint basePayment, uint baseDonation, uint callValue) constant retur
    function getMinimumEndowment(uint basePayment, uint baseDonation, uint callValue, uint requiredGa
    function isKnownCall(address callAddress) constant returns (bool);
    function getFirstSchedulableBlock() constant returns (uint);
    function getMinimumStackCheck() constant returns (uint16);
    function getMaximumStackCheck() constant returns (uint16);
    function getDefaultStackCheck() constant returns (uint16);
    function getDefaultRequiredGas() constant returns (uint);
    function getDefaultGracePeriod() constant returns (uint8);


    /*
     *  Next Call API
     */
    function getCallWindowSize() constant returns (uint);
    function getNextCall(uint blockNumber) constant returns (bytes32);
    function getNextCallSibling(address callAddress) constant returns (bytes32);
}
```

9.1.2 Abstract Call Contract Source Code

The following abstract solidity contract can be used to interact with a call contract from a solidity contract.

```
contract CallContractAPI {
    bytes public callData;
    address public contractAddress;
    uint8 public gracePeriod;
    address public schedulerAddress;
    uint public requiredGas;
    bool public isCancelled;
    bool public wasCalled;
    bool public wasSuccessful;
    uint public anchorGasPrice;
    uint public basePayment;
    bytes4 public abiSignature;
    uint public baseFee;
    uint public targetBlock;
    uint16 public requiredStackDepth;

    function execute() public;
    function cancel() public;

    function claim() public;

    address public claimer;
    uint public claimerDeposit;
    uint public claimAmount;

    function checkExecutionAuthorization(address executor, uint256 block_number) public returns (boo

    function getClaimAmountForBlock() public returns (uint);
    function getClaimAmountForBlock(uint256 block_number) public returns (uint);

    function registerData() public;
}
```

9.1.3 Only use what you need

The contracts above have stub functions for every API exposed by Alarm and CallerPool. It is safe to remove any functions or events from the abstract contracts that you do not intend to use.

9.2 Contract ABI

If you would like to interact with these contracts either from the javascript console, or the Ethereum wallet, you can use the following contract ABI.

9.2.1 Scheduler ABI

```
[
    {
        "constant": false,
        "inputs": [
            {
                "name": "contractAddress",
                "type": "address"
            },
            {
                "name": "abiSignature",
                "type": "bytes4"
            },
            {
                "name": "targetBlock",
                "type": "uint256"
            }
        ],
        "name": "scheduleCall",
        "outputs": [
            {
                "name": "",
                "type": "address"
            }
        ],
        "type": "function"
    },
    {
        "constant": false,
        "inputs": [
            {
                "name": "contractAddress",
                "type": "address"
            },
            {
                "name": "callValue",
                "type": "uint256"
            },
            {
                "name": "abiSignature",
                "type": "bytes4"
            },
            {
                "name": "targetBlock",
                "type": "uint256"
            },
            {
                "name": "requiredGas",
                "type": "uint256"
            },
            {
                "name": "gracePeriod",
                "type": "uint8"
            },
            {
                "name": "basePayment",
                "type": "uint256"
            }
        ],
        "name": "scheduleCall",
        "outputs": [
            {
                "name": "",
                "type": "address"
            }
        ],
        "type": "function"
    },
    {
        "constant": false,
        "inputs": [
            {
                "name": "contractAddress",
                "type": "address"
            }
        ],
        "name": "scheduleCall",
        "outputs": [
            {
                "name": "",
                "type": "address"
            }
        ],
        "type": "function"
    },
    {
        "constant": false,
        "inputs": [
            {
                "name": "contractAddress",
                "type": "address"
            },
            {
                "name": "abiSignature",
                "type": "bytes4"
            },
            {
                "name": "targetBlock",
                "type": "uint256"
            },
            {
                "name": "requiredGas",
                "type": "uint256"
            },
            {
                "name": "gracePeriod",
                "type": "uint8"
            }
        ],
        "name": "scheduleCall",
        "outputs": [
            {
                "name": "",
                "type": "address"
            }
        ],
        "type": "function"
    },
    {
        "constant": false,
        "inputs": [
            {
                "name": "contractAddress",
                "type": "address"
            },
            {
                "name": "abiSignature",
                "type": "bytes4"
            },
            {
                "name": "callData",
                "type": "bytes"
            },
            {
                "name": "requiredStackDepth",
                "type": "uint16"
            },
            {
                "name": "gracePeriod",
                "type": "uint8"
            },
            {
                "name": "args",
                "type": "uint256[5]"
            }
        ],
        "name": "scheduleCall",
        "outputs": [
            {
                "name": "",
                "type": "address"
            }
        ],
        "type": "function"
    },
    {
        "constant": false,
        "inputs": [
            {
                "name": "abiSignature",
                "type": "bytes4"
            },
            {
                "name": "callData",
                "type": "bytes"
            }
        ],
        "name": "scheduleCall",
        "outputs": [
            {
                "name": "",
                "type": "address"
            }
        ],
        "type": "function"
    },
    {
        "constant": false,
        "inputs": [
            {
                "name": "targetBlock",
                "type": "uint256"
            }
        ],
        "name": "scheduleCall",
        "outputs": [
            {
                "name": "",
                "type": "address"
            }
        ],
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [],
        "name": "getDefaultStackCheck",
        "outputs": [
            {
                "name": "",
                "type": "uint16"
            }
        ],
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [],
        "name": "getMinimumEndowment",
        "outputs": [
            {
                "name": "",
                "type": "uint256"
            }
        ],
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [],
        "name": "getMaximumCallGas",
        "outputs": [
            {
                "name": "",
                "type": "uint256"
            }
        ],
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [],
        "name": "callAPIVersion",
        "outputs": [
            {
                "name": "",
                "type": "uint256"
            }
        ],
        "type": "function"
    },
    {
        "constant": false,
        "inputs": [
            {
                "name": "contractAddress",
                "type": "address"
            },
            {
                "name": "abiSignature",
                "type": "bytes4"
            },
            {
                "name": "callValue",
                "type": "uint256"
            },
            {
                "name": "callData",
                "type": "bytes"
            },
            {
                "name": "targetBlock",
                "type": "uint256"
            }
        ],
        "name": "scheduleCall",
        "outputs": [
            {
                "name": "",
                "type": "address"
            }
        ],
        "type": "function"
    },
    {
        "constant": false,
        "inputs": [
            {
                "name": "contractAddress",
                "type": "address"
            },
            {
                "name": "abiSignature",
                "type": "bytes4"
            }
        ],
        "name": "scheduleCall",
        "outputs": [
            {
                "name": "",
                "type": "address"
            }
        ],
        "type": "function"
    },
    {
        "constant": false,
        "inputs": [
            {
                "name": "abiSignature",
                "type": "bytes4"
            },
            {
                "name": "targetBlock",
                "type": "uint256"
            },
            {
                "name": "requiredGas",
                "type": "uint256"
            },
            {
                "name": "gracePeriod",
                "type": "uint8"
            }
        ],
        "name": "scheduleCall",
        "outputs": [
            {
                "name": "",
                "type": "address"
            }
        ],
        "type": "function"
    },
    {
        "constant": false,
        "inputs": [
            {
                "name": "contractAddress",
                "type": "address"
            },
            {
                "name": "callValue",
                "type": "uint256"
            },
            {
                "name": "abiSignature",
                "type": "bytes4"
            }
        ],
        "name": "scheduleCall",
        "outputs": [
            {
                "name": "",
                "type": "address"
            }
        ],
        "type": "function"
    },
    {
        "constant": false,
        "inputs": [
            {
                "name": "abiSignature",
                "type": "bytes4"
            },
            {
                "name": "callData",
                "type": "bytes"
            },
            {
                "name": "targetBlock",
                "type": "uint256"
            }
        ],
        "name": "scheduleCall",
        "outputs": [
            {
                "name": "",
                "type": "address"
            }
        ],
        "type": "function"
    },
    {
        "constant": false,
        "inputs": [
            {
                "name": "contractAddress",
                "type": "address"
            },
            {
                "name": "abiSignature",
                "type": "bytes4"
            },
            {
                "name": "targetBlock",
                "type": "uint256"
            },
            {
                "name": "requiredGas",
                "type": "uint256"
            }
        ],
        "name": "scheduleCall",
        "outputs": [
            {
                "name": "",
                "type": "address"
            }
        ],
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [
            {
                "name": "callAddress",
                "type": "address"
            }
        ],
        "name": "getNextCallSibling",
        "outputs": [
            {
                "name": "",
                "type": "address"
            }
        ],
        "type": "function"
    },
    {
        "constant": false,
        "inputs": [
            {
                "name": "contractAddress",
                "type": "address"
            },
            {
                "name": "abiSignature",
                "type": "bytes4"
            },
            {
                "name": "callData",
                "type": "bytes"
            },
            {
                "name": "targetBlock",
                "type": "uint256"
            }
        ],
        "name": "scheduleCall",
        "outputs": [
            {
                "name": "",
                "type": "address"
            }
        ],
        "type": "function"
    },
    {
        "constant": false,
        "inputs": [
            {
                "name": "contractAddress",
                "type": "address"
            },
            {
                "name": "abiSignature",
                "type": "bytes4"
            },
            {
                "name": "callData",
                "type": "bytes"
            },
            {
                "name": "targetBlock",
                "type": "uint256"
            },
            {
                "name": "requiredGas",
                "type": "uint256"
            },
            {
                "name": "gracePeriod",
                "type": "uint8"
            },
            {
                "name": "basePayment",
                "type": "uint256"
            }
        ],
        "name": "scheduleCall",
        "outputs": [
            {
                "name": "",
                "type": "address"
            }
        ],
        "type": "function"
    },
    {
        "constant": false,
        "inputs": [
            {
                "name": "abiSignature",
                "type": "bytes4"
            }
        ],
        "name": "scheduleCall",
        "outputs": [
            {
                "name": "",
                "type": "address"
            }
        ],
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [
            {
                "name": "callAddress",
                "type": "address"
            }
        ],
        "name": "isKnownCall",
        "outputs": [
            {
                "name": "",
                "type": "bool"
            }
        ],
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [],
        "name": "getMaximumStackCheck",
        "outputs": [
            {
                "name": "",
                "type": "uint16"
            }
        ],
        "type": "function"
    },
    {
        "constant": false,
        "inputs": [
            {
                "name": "abiSignature",
                "type": "bytes4"
            },
            {
                "name": "callData",
                "type": "bytes"
            },
            {
                "name": "targetBlock",
                "type": "uint256"
            },
            {
                "name": "requiredGas",
                "type": "uint256"
            }
        ],
        "name": "scheduleCall",
        "outputs": [
            {
                "name": "",
                "type": "address"
            }
        ],
        "type": "function"
    },
    {
        "constant": false,
        "inputs": [
            {
                "name": "contractAddress",
                "type": "address"
            },
            {
                "name": "callValue",
                "type": "uint256"
            },
            {
                "name": "abiSignature",
                "type": "bytes4"
            },
            {
                "name": "targetBlock",
                "type": "uint256"
            },
            {
                "name": "requiredGas",
                "type": "uint256"
            },
            {
                "name": "gracePeriod",
                "type": "uint8"
            }
        ],
        "name": "scheduleCall",
        "outputs": [
            {
                "name": "",
                "type": "address"
            }
        ],
        "type": "function"
    },
    {
        "constant": false,
        "inputs": [
            {
                "name": "callData",
                "type": "bytes"
            }
        ],
        "name": "scheduleCall",
        "outputs": [
            {
                "name": "",
                "type": "address"
            }
        ],
        "type": "function"
    },
    {
        "constant": false,
        "inputs": [
            {
                "name": "contractAddress",
                "type": "address"
            },
            {
                "name": "abiSignature",
                "type": "bytes4"
            },
            {
                "name": "targetBlock",
                "type": "uint256"
            },
            {
                "name": "requiredGas",
                "type": "uint256"
            },
            {
                "name": "gracePeriod",
                "type": "uint8"
            },
            {
                "name": "basePayment",
                "type": "uint256"
            }
        ],
        "name": "scheduleCall",
        "outputs": [
            {
                "name": "",
                "type": "address"
            }
        ],
        "type": "function"
    },
    {
        "constant": false,
        "inputs": [
            {
                "name": "contractAddress",
                "type": "address"
            },
            {
                "name": "abiSignature",
                "type": "bytes4"
            },
            {
                "name": "callValue",
                "type": "uint256"
            },
            {
                "name": "callData",
                "type": "bytes"
            }
        ],
        "name": "scheduleCall",
        "outputs": [
            {
                "name": "",
                "type": "address"
            }
        ],
        "type": "function"
    },
    {
        "constant": false,
        "inputs": [
            {
                "name": "contractAddress",
                "type": "address"
            },
            {
                "name": "abiSignature",
                "type": "bytes4"
            },
            {
                "name": "callData",
                "type": "bytes"
            },
            {
                "name": "gracePeriod",
                "type": "uint8"
            },
            {
                "name": "args",
                "type": "uint256[4]"
            }
        ],
        "name": "scheduleCall",
        "outputs": [
            {
                "name": "",
                "type": "address"
            }
        ],
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [
            {
                "name": "basePayment",
                "type": "uint256"
            },
            {
                "name": "baseDonation",
                "type": "uint256"
            },
            {
                "name": "callValue",
                "type": "uint256"
            }
        ],
        "name": "getMinimumEndowment",
        "outputs": [
            {
                "name": "",
                "type": "uint256"
            }
        ],
        "type": "function"
    },
    {
        "constant": false,
        "inputs": [
            {
                "name": "abiSignature",
                "type": "bytes4"
            },
            {
                "name": "targetBlock",
                "type": "uint256"
            },
            {
                "name": "requiredGas",
                "type": "uint256"
            }
        ],
        "name": "scheduleCall",
        "outputs": [
            {
                "name": "",
                "type": "address"
            }
        ],
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [],
        "name": "defaultPayment",
        "outputs": [
            {
                "name": "",
                "type": "uint256"
            }
        ],
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [],
        "name": "getDefaultGracePeriod",
        "outputs": [
            {
                "name": "",
                "type": "uint8"
            }
        ],
        "type": "function"
    },
    {
        "constant": false,
        "inputs": [
            {
                "name": "abiSignature",
                "type": "bytes4"
            },
            {
                "name": "callData",
                "type": "bytes"
            },
            {
                "name": "requiredStackDepth",
                "type": "uint16"
            },
            {
                "name": "gracePeriod",
                "type": "uint8"
            },
            {
                "name": "callValue",
                "type": "uint256"
            },
            {
                "name": "targetBlock",
                "type": "uint256"
            },
            {
                "name": "requiredGas",
                "type": "uint256"
            },
            {
                "name": "basePayment",
                "type": "uint256"
            },
            {
                "name": "baseDonation",
                "type": "uint256"
            }
        ],
        "name": "scheduleCall",
        "outputs": [
            {
                "name": "",
                "type": "address"
            }
        ],
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [],
        "name": "getMinimumStackCheck",
        "outputs": [
            {
                "name": "",
                "type": "uint16"
            }
        ],
        "type": "function"
    },
    {
        "constant": false,
        "inputs": [
            {
                "name": "contractAddress",
                "type": "address"
            },
            {
                "name": "abiSignature",
                "type": "bytes4"
            },
            {
                "name": "callData",
                "type": "bytes"
            },
            {
                "name": "targetBlock",
                "type": "uint256"
            },
            {
                "name": "requiredGas",
                "type": "uint256"
            }
        ],
        "name": "scheduleCall",
        "outputs": [
            {
                "name": "",
                "type": "address"
            }
        ],
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [],
        "name": "getMinimumCallGas",
        "outputs": [
            {
                "name": "",
                "type": "uint256"
            }
        ],
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [],
        "name": "getCallWindowSize",
        "outputs": [
            {
                "name": "",
                "type": "uint256"
            }
        ],
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [
            {
                "name": "blockNumber",
                "type": "uint256"
            }
        ],
        "name": "getNextCall",
        "outputs": [
            {
                "name": "",
                "type": "address"
            }
        ],
        "type": "function"
    },
    {
        "constant": false,
        "inputs": [
            {
                "name": "callValue",
                "type": "uint256"
            },
            {
                "name": "contractAddress",
                "type": "address"
            }
        ],
        "name": "scheduleCall",
        "outputs": [
            {
                "name": "",
                "type": "address"
            }
        ],
        "type": "function"
    },
    {
        "constant": false,
        "inputs": [
            {
                "name": "contractAddress",
                "type": "address"
            },
            {
                "name": "abiSignature",
                "type": "bytes4"
            },
            {
                "name": "callData",
                "type": "bytes"
            }
        ],
        "name": "scheduleCall",
        "outputs": [
            {
                "name": "",
                "type": "address"
            }
        ],
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [
            {
                "name": "basePayment",
                "type": "uint256"
            }
        ],
        "name": "getMinimumEndowment",
        "outputs": [
            {
                "name": "",
                "type": "uint256"
            }
        ],
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [],
        "name": "getFirstSchedulableBlock",
        "outputs": [
            {
                "name": "",
                "type": "uint256"
            }
        ],
        "type": "function"
    },
    {
        "constant": false,
        "inputs": [
            {
                "name": "abiSignature",
                "type": "bytes4"
            },
            {
                "name": "callData",
                "type": "bytes"
            },
            {
                "name": "targetBlock",
                "type": "uint256"
            },
            {
                "name": "requiredGas",
                "type": "uint256"
            },
            {
                "name": "gracePeriod",
                "type": "uint8"
            },
            {
                "name": "basePayment",
                "type": "uint256"
            }
        ],
        "name": "scheduleCall",
        "outputs": [
            {
                "name": "",
                "type": "address"
            }
        ],
        "type": "function"
    },
    {
        "constant": false,
        "inputs": [
            {
                "name": "contractAddress",
                "type": "address"
            },
            {
                "name": "targetBlock",
                "type": "uint256"
            }
        ],
        "name": "scheduleCall",
        "outputs": [
            {
                "name": "",
                "type": "address"
            }
        ],
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [],
        "name": "getDefaultDonation",
        "outputs": [
            {
                "name": "",
                "type": "uint256"
            }
        ],
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [],
        "name": "getMinimumGracePeriod",
        "outputs": [
            {
                "name": "",
                "type": "uint256"
            }
        ],
        "type": "function"
    },
    {
        "constant": false,
        "inputs": [],
        "name": "scheduleCall",
        "outputs": [
            {
                "name": "",
                "type": "address"
            }
        ],
        "type": "function"
    },
    {
        "constant": false,
        "inputs": [
            {
                "name": "abiSignature",
                "type": "bytes4"
            },
            {
                "name": "targetBlock",
                "type": "uint256"
            }
        ],
        "name": "scheduleCall",
        "outputs": [
            {
                "name": "",
                "type": "address"
            }
        ],
        "type": "function"
    },
    {
        "constant": false,
        "inputs": [
            {
                "name": "contractAddress",
                "type": "address"
            },
            {
                "name": "targetBlock",
                "type": "uint256"
            },
            {
                "name": "callValue",
                "type": "uint256"
            }
        ],
        "name": "scheduleCall",
        "outputs": [
            {
                "name": "",
                "type": "address"
            }
        ],
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [],
        "name": "getDefaultRequiredGas",
        "outputs": [
            {
                "name": "",
                "type": "uint256"
            }
        ],
        "type": "function"
    },
    {
        "constant": false,
        "inputs": [
            {
                "name": "abiSignature",
                "type": "bytes4"
            },
            {
                "name": "targetBlock",
                "type": "uint256"
            },
            {
                "name": "requiredGas",
                "type": "uint256"
            },
            {
                "name": "gracePeriod",
                "type": "uint8"
            },
            {
                "name": "basePayment",
                "type": "uint256"
            }
        ],
        "name": "scheduleCall",
        "outputs": [
            {
                "name": "",
                "type": "address"
            }
        ],
        "type": "function"
    },
    {
        "constant": false,
        "inputs": [],
        "name": "updateDefaultPayment",
        "outputs": [],
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [
            {
                "name": "basePayment",
                "type": "uint256"
            },
            {
                "name": "baseDonation",
                "type": "uint256"
            },
            {
                "name": "callValue",
                "type": "uint256"
            },
            {
                "name": "requiredGas",
                "type": "uint256"
            }
        ],
        "name": "getMinimumEndowment",
        "outputs": [
            {
                "name": "",
                "type": "uint256"
            }
        ],
        "type": "function"
    },
    {
        "constant": false,
        "inputs": [
            {
                "name": "contractAddress",
                "type": "address"
            },
            {
                "name": "abiSignature",
                "type": "bytes4"
            },
            {
                "name": "callData",
                "type": "bytes"
            },
            {
                "name": "targetBlock",
                "type": "uint256"
            },
            {
                "name": "requiredGas",
                "type": "uint256"
            },
            {
                "name": "gracePeriod",
                "type": "uint8"
            }
        ],
        "name": "scheduleCall",
        "outputs": [
            {
                "name": "",
                "type": "address"
            }
        ],
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [
            {
                "name": "basePayment",
                "type": "uint256"
            },
            {
                "name": "baseDonation",
                "type": "uint256"
            }
        ],
        "name": "getMinimumEndowment",
        "outputs": [
            {
                "name": "",
                "type": "uint256"
            }
        ],
        "type": "function"
    },
    {
        "inputs": [],
        "type": "constructor"
    }
]
```

9.2.2 Call Contract ABI

```
[
    {
        "constant": true,
        "inputs": [],
        "name": "wasSuccessful",
        "outputs": [
            {
                "name": "",
                "type": "bool"
            }
        ],
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [],
        "name": "targetBlock",
        "outputs": [
            {
                "name": "",
                "type": "uint256"
            }
        ],
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [],
        "name": "firstClaimBlock",
        "outputs": [
            {
                "name": "",
                "type": "uint256"
            }
        ],
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [],
        "name": "getExtraGas",
        "outputs": [
            {
                "name": "",
                "type": "uint256"
            }
        ],
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [
            {
                "name": "n",
                "type": "uint256"
            }
        ],
        "name": "__dig",
        "outputs": [
            {
                "name": "",
                "type": "bool"
            }
        ],
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [
            {
                "name": "executor",
                "type": "address"
            },
            {
                "name": "block_number",
                "type": "uint256"
            }
        ],
        "name": "checkExecutionAuthorization",
        "outputs": [
            {
                "name": "",
                "type": "bool"
            }
        ],
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [],
        "name": "requiredStackDepth",
        "outputs": [
            {
                "name": "",
                "type": "uint16"
            }
        ],
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [],
        "name": "callAPIVersion",
        "outputs": [
            {
                "name": "",
                "type": "uint256"
            }
        ],
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [],
        "name": "claimerDeposit",
        "outputs": [
            {
                "name": "",
                "type": "uint256"
            }
        ],
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [],
        "name": "anchorGasPrice",
        "outputs": [
            {
                "name": "",
                "type": "uint256"
            }
        ],
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [],
        "name": "isCancellable",
        "outputs": [
            {
                "name": "",
                "type": "bool"
            }
        ],
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [],
        "name": "callData",
        "outputs": [
            {
                "name": "",
                "type": "bytes"
            }
        ],
        "type": "function"
    },
    {
        "constant": false,
        "inputs": [],
        "name": "claim",
        "outputs": [
            {
                "name": "",
                "type": "bool"
            }
        ],
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [],
        "name": "getClaimAmountForBlock",
        "outputs": [
            {
                "name": "",
                "type": "uint256"
            }
        ],
        "type": "function"
    },
    {
        "constant": false,
        "inputs": [],
        "name": "execute",
        "outputs": [],
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [],
        "name": "baseDonation",
        "outputs": [
            {
                "name": "",
                "type": "uint256"
            }
        ],
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [],
        "name": "getOverhead",
        "outputs": [
            {
                "name": "",
                "type": "uint256"
            }
        ],
        "type": "function"
    },
    {
        "constant": false,
        "inputs": [
            {
                "name": "executor",
                "type": "address"
            },
            {
                "name": "startGas",
                "type": "uint256"
            }
        ],
        "name": "beforeExecute",
        "outputs": [
            {
                "name": "",
                "type": "bool"
            }
        ],
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [],
        "name": "claimAmount",
        "outputs": [
            {
                "name": "",
                "type": "uint256"
            }
        ],
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [],
        "name": "origin",
        "outputs": [
            {
                "name": "",
                "type": "address"
            }
        ],
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [],
        "name": "isCancelled",
        "outputs": [
            {
                "name": "",
                "type": "bool"
            }
        ],
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [],
        "name": "requiredGas",
        "outputs": [
            {
                "name": "",
                "type": "uint256"
            }
        ],
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [],
        "name": "gracePeriod",
        "outputs": [
            {
                "name": "",
                "type": "uint8"
            }
        ],
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [],
        "name": "lastClaimBlock",
        "outputs": [
            {
                "name": "",
                "type": "uint256"
            }
        ],
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [],
        "name": "schedulerAddress",
        "outputs": [
            {
                "name": "",
                "type": "address"
            }
        ],
        "type": "function"
    },
    {
        "constant": false,
        "inputs": [],
        "name": "registerData",
        "outputs": [
            {
                "name": "",
                "type": "bool"
            }
        ],
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [],
        "name": "state",
        "outputs": [
            {
                "name": "",
                "type": "uint8"
            }
        ],
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [],
        "name": "basePayment",
        "outputs": [
            {
                "name": "",
                "type": "uint256"
            }
        ],
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [],
        "name": "wasCalled",
        "outputs": [
            {
                "name": "",
                "type": "bool"
            }
        ],
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [],
        "name": "abiSignature",
        "outputs": [
            {
                "name": "",
                "type": "bytes4"
            }
        ],
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [],
        "name": "maxClaimBlock",
        "outputs": [
            {
                "name": "",
                "type": "uint256"
            }
        ],
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [],
        "name": "claimer",
        "outputs": [
            {
                "name": "",
                "type": "address"
            }
        ],
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [],
        "name": "callValue",
        "outputs": [
            {
                "name": "",
                "type": "uint256"
            }
        ],
        "type": "function"
    },
    {
        "constant": false,
        "inputs": [],
        "name": "cancel",
        "outputs": [],
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [
            {
                "name": "block_number",
                "type": "uint256"
            }
        ],
        "name": "getClaimAmountForBlock",
        "outputs": [
            {
                "name": "",
                "type": "uint256"
            }
        ],
        "type": "function"
    },
    {
        "constant": true,
        "inputs": [],
        "name": "contractAddress",
        "outputs": [
            {
                "name": "",
                "type": "address"
            }
        ],
        "type": "function"
    },
    {
        "inputs": [
            {
                "name": "_schedulerAddress",
                "type": "address"
            },
            {
                "name": "_targetBlock",
                "type": "uint256"
            },
            {
                "name": "_gracePeriod",
                "type": "uint8"
            },
            {
                "name": "_contractAddress",
                "type": "address"
            },
            {
                "name": "_abiSignature",
                "type": "bytes4"
            },
            {
                "name": "_callData",
                "type": "bytes"
            },
            {
                "name": "_callValue",
                "type": "uint256"
            },
            {
                "name": "_requiredGas",
                "type": "uint256"
            },
            {
                "name": "_requiredStackDepth",
                "type": "uint16"
            },
            {
                "name": "_basePayment",
                "type": "uint256"
            },
            {
                "name": "_baseDonation",
                "type": "uint256"
            }
        ],
        "type": "constructor"
    }
]
```

CHAPTER 10

Events

The following events are used to log notable events within the Alarm service.

10.1 Scheduler Events

The Scheduler contract logs the following events.

10.1.1 Call Scheduled

- **Solidity Event Signature:** `CallScheduled(address callAddress)`

Logged when a new scheduled call is created.

10.1.2 Call Rejected

- **Solidity Event Signature:** `CallRejected(address indexed schedulerAddress, bytes32 reason)`

Logged when an attempt to schedule a function call fails.

10.2 Call Contract Events

Each CallContract logs the following events.

10.2.1 Call Executed

- **Solidity Event Signature:** **`CallExecuted(address indexed executor, uint gasCost, uint payment, u`**
  ;

Executed when the call is executed.

10.2.2 Call Aborted

- **Solidity Event Signature:** `_CallAborted(address executor, bytes32 reason)`

Executed when an attempt is made to execute a scheduled call is rejected. The `reason` value in this log entry contains a short string representation of why the call was rejected. (Note that this event name starts with an underscore)

Reasons:

- `NOT_ENOUGH_GAS` - Executing transaction less than the `requiredGas` value
- `ALREADY_CALLED` - The call has already been executed.
- `NOT_IN_CALL_WINDOW` - The transaction is occurring outside of the call window.
- `NOT_AUTHORIZED` - Attempting to execute a claimed call during the period that the claimer has exclusive call rights. * `STACK_TOO_DEEP`
- The stack depth could not be extended to `requiredStackDepth`.

CHAPTER 11

Terminology

Definitions for various terms that are used regularly to describe parts of the Alarm service.

11.1 General

Ethereum Alarm Clock, Alarm, Alarm Service Generic terms for the service as a whole.

Scheduler Contract The solidity contract responsible for scheduling a function call.

Call Contract The solidity contract that is deployed for each scheduled call. This contract handles execution of the call, registration of call data, gas reimbursment, and payment and fee dispursment.

Scheduled Call A contract function call that has been registered with the Alarm service to be executed at a specified time in the future (currently denoted by a block number).

11.2 Calls and Call Scheduling

Scheduler The account which scheduled the function call.

Executor The account which initiates the transaction which executes a scheduled function call.

Target Block The first block number that a scheduled call can be called.

Grace Period The number of blocks after the **target block** that a scheduled call can be be called.

Freeze Window The 10 blocks directly preceeding the target block for a call

Claim Window The 255 block window prior to the Freeze Window during which the call may be claimed for exclusive rights to execution during the first 16 blocks of the call window.

First Claim Block The first block in the 255 block claim window.

Max Claim Block The 240th block of the claim window. This is the block when the value of the call if claimed is equal to the `basePayment` for thec all.

Call Window Used to refer to either the full window of blocks during which a scheduled call can be executed, or a portion of this window that has been designated to a specific caller.

Payment The amount that is paid to the executor of the scheduled call.

Donation The amount that is paid to the creator of the Alarm service.

Anchor Gas Price The gas price that was used when scheduling the scheduled call.

Gas Price Scalar A number ranging from 0 - 2 that is derived from the difference between the gas price of the executing transaction and the **anchor gas grice**. This number equals 1 when the two numbers are equal. It approaches 2 as the executing gas grice drops below the anchor gas price. It approaches zero as the executing gas price rises above the anchor gas price. This multiplier is applied to the payment and fee values, intending to motivate the executor to use a reasonable gas price.

CHAPTER 12

Changelog

12.1 0.6.0

- Scheduled calls can now specify a required gas amount. This takes place of the `suggestedGas` api from 0.6.0
- Scheduled calls can now send value along with the transaction.
- Calls now protect against stack depth attacks. This is configurable via the `requiredStackDepth` option.
- Calls can now be scheduled as soon as 10 blocks in the future.
- Experimental implementation of market-based value for the `defaultPayment`
- `scheduleCall` now has 31 different call signatures.

12.2 0.6.0

- Each scheduled call now exists as it's own contract, referred to as a call contract.
- Removal of the Caller Pool
- Introduction of the claim api for call.
- Call Portability. Scheduled calls can now be trustlessly imported into future versions of the service.

12.3 0.5.0

- Each scheduled call now exists as it's own contract, referred to as a call contract.
- The authorization API has been removed. It is now possible for the contract being called to look up `msg.sender` on the scheduling contract and find out who scheduled the call.
- The account management API has been removed. Each call contract now manages it's own gas money, the remainder of which is given back to the scheduler after the call is executed.
- All of the information that used to be stored about the call execution is now placed in event logs (gasUsed, wasSuccessful, wasCalled, etc)

12.4 0.4.0

- Convert Alarm service to use library contracts for all functionality.
- CallerPool contract API is now integrated into the Alarm API

12.5 0.3.0

- Convert Alarm service to use Grove for tracking scheduled call ordering.
- Enable logging most notable Alarm service events.
- Two additional convenience functions for invoking `scheduleCall` with **gracePeriod** and **nonce** as optional parameters.

12.6 0.2.0

- Fix for Issue 42. Make the free-for-all bond bonus restrict itself to the correct set of callers.
- Re-enable the right tree rotation in favor of removing three `getLastX` function. This is related to the pi-million gas limit which is restricting the code size of the contract.

12.7 0.1.0

- Initial release.