```diff
diff --git a/.gitignore b/.gitignore
index 80f9dd1..1612964 100644
--- a/.gitignore
+++ b/.gitignore
@@ -40,7 +40,6 @@ db*
 *.fdb*
 *.fls
 *.gz
-*.pdf
 
 .coverage
 
diff --git a/accounts/accounts.go b/accounts/accounts.go
index dd7df0b..00420ca 100644
--- a/accounts/accounts.go
+++ b/accounts/accounts.go
@@ -31,10 +31,10 @@ import (
        "fmt"
        "math/big"
 
-       "github.com/ava-labs/coreth/core/types"
-       "github.com/ava-labs/coreth/interfaces"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/event"
+       "gitlab.com/flarenetwork/coreth/core/types"
+       "gitlab.com/flarenetwork/coreth/interfaces"
        "golang.org/x/crypto/sha3"
 )
 
diff --git a/accounts/external/backend.go b/accounts/external/backend.go
index 558700f..8302c0c 100644
--- a/accounts/external/backend.go
+++ b/accounts/external/backend.go
@@ -31,15 +31,15 @@ import (
        "math/big"
        "sync"
 
-       "github.com/ava-labs/coreth/accounts"
-       "github.com/ava-labs/coreth/core/types"
-       "github.com/ava-labs/coreth/interfaces"
-       "github.com/ava-labs/coreth/rpc"
-       "github.com/ava-labs/coreth/signer/core/apitypes"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/common/hexutil"
        "github.com/ethereum/go-ethereum/event"
        "github.com/ethereum/go-ethereum/log"
+       "gitlab.com/flarenetwork/coreth/accounts"
+       "gitlab.com/flarenetwork/coreth/core/types"
+       "gitlab.com/flarenetwork/coreth/interfaces"
+       "gitlab.com/flarenetwork/coreth/rpc"
+       "gitlab.com/flarenetwork/coreth/signer/core/apitypes"
 )
 
 type ExternalBackend struct {
diff --git a/accounts/keystore/account_cache.go b/accounts/keystore/account_cache.go
index 4c35aa7..fbe0a89 100644
--- a/accounts/keystore/account_cache.go
+++ b/accounts/keystore/account_cache.go
@@ -37,10 +37,10 @@ import (
        "sync"
        "time"
 
-       "github.com/ava-labs/coreth/accounts"
        mapset "github.com/deckarep/golang-set"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/log"
+       "gitlab.com/flarenetwork/coreth/accounts"
 )
 
 // Minimum amount of time between cache reloads. This limit applies if the platform does
diff --git a/accounts/keystore/account_cache_test.go b/accounts/keystore/account_cache_test.go
index 3d341d5..81f80d6 100644
--- a/accounts/keystore/account_cache_test.go
+++ b/accounts/keystore/account_cache_test.go
@@ -37,10 +37,10 @@ import (
        "testing"
        "time"
 
-       "github.com/ava-labs/coreth/accounts"
        "github.com/cespare/cp"
        "github.com/davecgh/go-spew/spew"
        "github.com/ethereum/go-ethereum/common"
+       "gitlab.com/flarenetwork/coreth/accounts"
 )
 
 var (
diff --git a/accounts/keystore/key.go b/accounts/keystore/key.go
index 71402d3..ad6cc6c 100644
--- a/accounts/keystore/key.go
+++ b/accounts/keystore/key.go
@@ -39,10 +39,10 @@ import (
        "strings"
        "time"
 
-       "github.com/ava-labs/coreth/accounts"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/crypto"
        "github.com/google/uuid"
+       "gitlab.com/flarenetwork/coreth/accounts"
 )
 
 const (
diff --git a/accounts/keystore/keystore.go b/accounts/keystore/keystore.go
index ff82ef8..e1c177d 100644
--- a/accounts/keystore/keystore.go
+++ b/accounts/keystore/keystore.go
@@ -42,11 +42,11 @@ import (
        "sync"
        "time"
 
-       "github.com/ava-labs/coreth/accounts"
-       "github.com/ava-labs/coreth/core/types"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/crypto"
        "github.com/ethereum/go-ethereum/event"
+       "gitlab.com/flarenetwork/coreth/accounts"
+       "gitlab.com/flarenetwork/coreth/core/types"
 )
 
 var (
diff --git a/accounts/keystore/keystore_test.go b/accounts/keystore/keystore_test.go
index 651ab70..70fbeb7 100644
--- a/accounts/keystore/keystore_test.go
+++ b/accounts/keystore/keystore_test.go
@@ -38,10 +38,10 @@ import (
        "testing"
        "time"
 
-       "github.com/ava-labs/coreth/accounts"
```

```diff
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/crypto"
        "github.com/ethereum/go-ethereum/event"
+       "gitlab.com/flarenetwork/coreth/accounts"
 )

 var testSigData = make([]byte, 32)
diff --git a/accounts/keystore/passphrase.go b/accounts/keystore/passphrase.go
index 5da41f6..73f26d2 100644
--- a/accounts/keystore/passphrase.go
+++ b/accounts/keystore/passphrase.go
@@ -48,11 +48,11 @@ import (
        "os"
        "path/filepath"

-       "github.com/ava-labs/coreth/accounts"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/common/math"
        "github.com/ethereum/go-ethereum/crypto"
        "github.com/google/uuid"
+       "gitlab.com/flarenetwork/coreth/accounts"
        "golang.org/x/crypto/pbkdf2"
        "golang.org/x/crypto/scrypt"
 )
diff --git a/accounts/keystore/presale.go b/accounts/keystore/presale.go
index 1dfbd9c..64f020d 100644
--- a/accounts/keystore/presale.go
+++ b/accounts/keystore/presale.go
@@ -35,9 +35,9 @@ import (
        "errors"
        "fmt"

-       "github.com/ava-labs/coreth/accounts"
        "github.com/ethereum/go-ethereum/crypto"
        "github.com/google/uuid"
+       "gitlab.com/flarenetwork/coreth/accounts"
        "golang.org/x/crypto/pbkdf2"
 )

diff --git a/accounts/keystore/wallet.go b/accounts/keystore/wallet.go
index 78eac1d..e603ac5 100644
--- a/accounts/keystore/wallet.go
+++ b/accounts/keystore/wallet.go
@@ -29,11 +29,11 @@ package keystore
 import (
        "math/big"

-       "github.com/ava-labs/coreth/interfaces"
+       "gitlab.com/flarenetwork/coreth/interfaces"

-       "github.com/ava-labs/coreth/accounts"
-       "github.com/ava-labs/coreth/core/types"
        "github.com/ethereum/go-ethereum/crypto"
+       "gitlab.com/flarenetwork/coreth/accounts"
+       "gitlab.com/flarenetwork/coreth/core/types"
 )

 // keystoreWallet implements the accounts.Wallet interface for the original
diff --git a/accounts/scwallet/hub.go b/accounts/scwallet/hub.go
index 7a630fa..c05ba6f 100644
--- a/accounts/scwallet/hub.go
+++ b/accounts/scwallet/hub.go
@@ -51,11 +51,11 @@ import (
        "sync"
        "time"

-       "github.com/ava-labs/coreth/accounts"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/event"
        "github.com/ethereum/go-ethereum/log"
        pcsc "github.com/gballet/go-libpcsclite"
+       "gitlab.com/flarenetwork/coreth/accounts"
 )

 // Scheme is the URI prefix for smartcard wallets.
diff --git a/accounts/scwallet/wallet.go b/accounts/scwallet/wallet.go
index fcecc10..b912e21 100644
--- a/accounts/scwallet/wallet.go
+++ b/accounts/scwallet/wallet.go
@@ -43,14 +43,14 @@ import (
        "sync"
        "time"

-       "github.com/ava-labs/coreth/accounts"
-       "github.com/ava-labs/coreth/core/types"
-       "github.com/ava-labs/coreth/interfaces"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/crypto"
        "github.com/ethereum/go-ethereum/log"
        pcsc "github.com/gballet/go-libpcsclite"
        "github.com/status-im/keycard-go/derivationpath"
+       "gitlab.com/flarenetwork/coreth/accounts"
+       "gitlab.com/flarenetwork/coreth/core/types"
+       "gitlab.com/flarenetwork/coreth/interfaces"
 )

 // ErrPairingPasswordNeeded is returned if opening the smart card requires pairing with a pairing
diff --git a/chain/chain_test.go b/chain/chain_test.go
index 2d8c7e9..9057d9d 100644
--- a/chain/chain_test.go
+++ b/chain/chain_test.go
@@ -9,20 +9,20 @@ import (
        "math/rand"
        "testing"

-       "github.com/ava-labs/coreth/accounts/keystore"
-       "github.com/ava-labs/coreth/consensus/dummy"
-       "github.com/ava-labs/coreth/core"
-       "github.com/ava-labs/coreth/core/rawdb"
-       "github.com/ava-labs/coreth/core/state"
-       "github.com/ava-labs/coreth/core/types"
-       "github.com/ava-labs/coreth/eth"
-       "github.com/ava-labs/coreth/eth/ethconfig"
-       "github.com/ava-labs/coreth/node"
-       "github.com/ava-labs/coreth/params"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/common/hexutil"
        "github.com/ethereum/go-ethereum/log"
        "github.com/ethereum/go-ethereum/rlp"
+       "gitlab.com/flarenetwork/coreth/accounts/keystore"
+       "gitlab.com/flarenetwork/coreth/consensus/dummy"
+       "gitlab.com/flarenetwork/coreth/core"
+       "gitlab.com/flarenetwork/coreth/core/rawdb"
+       "gitlab.com/flarenetwork/coreth/core/state"
+       "gitlab.com/flarenetwork/coreth/core/types"
+       "gitlab.com/flarenetwork/coreth/eth"
+       "gitlab.com/flarenetwork/coreth/eth/ethconfig"
+       "gitlab.com/flarenetwork/coreth/node"
+       "gitlab.com/flarenetwork/coreth/params"
 )
 type testChain struct {
diff --git a/chain/coreth.go b/chain/coreth.go
index 95018be..bc107de 100644
--- a/chain/coreth.go
+++ b/chain/coreth.go
@@ -7,15 +7,15 @@ import (
        "fmt"
        "time"
 
-       "github.com/ava-labs/coreth/consensus/dummy"
-       "github.com/ava-labs/coreth/core"
-       "github.com/ava-labs/coreth/core/state"
-       "github.com/ava-labs/coreth/core/types"
-       "github.com/ava-labs/coreth/eth"
-       "github.com/ava-labs/coreth/node"
-       "github.com/ava-labs/coreth/rpc"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/ethdb"
+       "gitlab.com/flarenetwork/coreth/consensus/dummy"
+       "gitlab.com/flarenetwork/coreth/core"
+       "gitlab.com/flarenetwork/coreth/core/state"
+       "gitlab.com/flarenetwork/coreth/core/types"
+       "gitlab.com/flarenetwork/coreth/eth"
+       "gitlab.com/flarenetwork/coreth/node"
+       "gitlab.com/flarenetwork/coreth/rpc"
 )
 
 var (
diff --git a/chain/counter_test.go b/chain/counter_test.go
index 7209ac5..4fd88fe 100644
--- a/chain/counter_test.go
+++ b/chain/counter_test.go
@@ -15,8 +15,8 @@ import (
 
        "testing"
 
-       "github.com/ava-labs/coreth/core/types"
        "github.com/ethereum/go-ethereum/common"
+       "gitlab.com/flarenetwork/coreth/core/types"
 
        "github.com/ethereum/go-ethereum/log"
 )
diff --git a/chain/multicoin_test.go b/chain/multicoin_test.go
index a2ce4cb..f40a678 100644
--- a/chain/multicoin_test.go
+++ b/chain/multicoin_test.go
@@ -28,19 +28,19 @@ import (
        "strings"
        "testing"
 
-       "github.com/ava-labs/coreth/accounts/keystore"
-       "github.com/ava-labs/coreth/consensus/dummy"
-       "github.com/ava-labs/coreth/core"
-       "github.com/ava-labs/coreth/core/rawdb"
-       "github.com/ava-labs/coreth/core/types"
-       "github.com/ava-labs/coreth/core/vm"
-       "github.com/ava-labs/coreth/eth"
-       "github.com/ava-labs/coreth/eth/ethconfig"
-       "github.com/ava-labs/coreth/node"
        "github.com/ethereum/go-ethereum/accounts/abi"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/crypto"
        "github.com/ethereum/go-ethereum/log"
+       "gitlab.com/flarenetwork/coreth/accounts/keystore"
+       "gitlab.com/flarenetwork/coreth/consensus/dummy"
+       "gitlab.com/flarenetwork/coreth/core"
+       "gitlab.com/flarenetwork/coreth/core/rawdb"
+       "gitlab.com/flarenetwork/coreth/core/types"
+       "gitlab.com/flarenetwork/coreth/core/vm"
+       "gitlab.com/flarenetwork/coreth/eth"
+       "gitlab.com/flarenetwork/coreth/eth/ethconfig"
+       "gitlab.com/flarenetwork/coreth/node"
 )
 
 // TestMulticoin tests multicoin low-level state management and regular
diff --git a/chain/payment_test.go b/chain/payment_test.go
index 5d9da4d..f17e260 100644
--- a/chain/payment_test.go
+++ b/chain/payment_test.go
@@ -7,8 +7,8 @@ import (
        "math/big"
        "testing"
 
-       "github.com/ava-labs/coreth/core/types"
        "github.com/ethereum/go-ethereum/log"
+       "gitlab.com/flarenetwork/coreth/core/types"
 )
 
 // TestPayment tests basic payment (balance, not multi-coin)
diff --git a/chain/subscribe_accepted_heads_test.go b/chain/subscribe_accepted_heads_test.go
index 0a94bfd..ab7315e 100644
--- a/chain/subscribe_accepted_heads_test.go
+++ b/chain/subscribe_accepted_heads_test.go
@@ -4,10 +4,10 @@ import (
        "math/big"
        "testing"
 
-       "github.com/ava-labs/coreth/core"
-       "github.com/ava-labs/coreth/core/types"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/log"
+       "gitlab.com/flarenetwork/coreth/core"
+       "gitlab.com/flarenetwork/coreth/core/types"
 )
 
 func TestAcceptedHeadSubscriptions(t *testing.T) {
diff --git a/chain/subscribe_block_logs_test.go b/chain/subscribe_block_logs_test.go
index 26661da..2b2d2ab 100644
--- a/chain/subscribe_block_logs_test.go
+++ b/chain/subscribe_block_logs_test.go
@@ -6,10 +6,10 @@ import (
        "testing"
        "time"
 
-       "github.com/ava-labs/coreth/eth/filters"
+       "gitlab.com/flarenetwork/coreth/eth/filters"
 
-       "github.com/ava-labs/coreth/core/types"
        "github.com/ethereum/go-ethereum/common"
+       "gitlab.com/flarenetwork/coreth/core/types"
 )
 
 func TestBlockLogsAllowUnfinalized(t *testing.T) {
diff --git a/chain/subscribe_transactions_test.go b/chain/subscribe_transactions_test.go
index aac6db4..9388dae 100644
--- a/chain/subscribe_transactions_test.go
+++ b/chain/subscribe_transactions_test.go
@@ -4,10 +4,10 @@ import (
        "math/big"
        "testing"

-       "github.com/ava-labs/coreth/eth/filters"
+       "gitlab.com/flarenetwork/coreth/eth/filters"

-       "github.com/ava-labs/coreth/core/types"
        "github.com/ethereum/go-ethereum/common"
+       "gitlab.com/flarenetwork/coreth/core/types"
 )

 func TestSubscribeTransactions(t *testing.T) {
diff --git a/chain/test_chain.go b/chain/test_chain.go
index e4420de..3d71d82 100644
--- a/chain/test_chain.go
+++ b/chain/test_chain.go
@@ -8,17 +8,17 @@ import (
        "math/big"
        "testing"

-       "github.com/ava-labs/coreth/accounts/keystore"
-       "github.com/ava-labs/coreth/consensus/dummy"
-       "github.com/ava-labs/coreth/core"
-       "github.com/ava-labs/coreth/core/rawdb"
-       "github.com/ava-labs/coreth/core/types"
-       "github.com/ava-labs/coreth/eth"
-       "github.com/ava-labs/coreth/eth/ethconfig"
-       "github.com/ava-labs/coreth/node"
-       "github.com/ava-labs/coreth/params"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/common/hexutil"
+       "gitlab.com/flarenetwork/coreth/accounts/keystore"
+       "gitlab.com/flarenetwork/coreth/consensus/dummy"
+       "gitlab.com/flarenetwork/coreth/core"
+       "gitlab.com/flarenetwork/coreth/core/rawdb"
+       "gitlab.com/flarenetwork/coreth/core/types"
+       "gitlab.com/flarenetwork/coreth/eth"
+       "gitlab.com/flarenetwork/coreth/eth/ethconfig"
+       "gitlab.com/flarenetwork/coreth/node"
+       "gitlab.com/flarenetwork/coreth/params"
 )

 var (
diff --git a/consensus/consensus.go b/consensus/consensus.go
index 6bc13a8..af462ca 100644
--- a/consensus/consensus.go
+++ b/consensus/consensus.go
@@ -30,11 +30,11 @@ package consensus
 import (
        "math/big"

-       "github.com/ava-labs/coreth/core/state"
-       "github.com/ava-labs/coreth/core/types"
-       "github.com/ava-labs/coreth/params"
-       "github.com/ava-labs/coreth/rpc"
        "github.com/ethereum/go-ethereum/common"
+       "gitlab.com/flarenetwork/coreth/core/state"
+       "gitlab.com/flarenetwork/coreth/core/types"
+       "gitlab.com/flarenetwork/coreth/params"
+       "gitlab.com/flarenetwork/coreth/rpc"
 )

 // ChainHeaderReader defines a small collection of methods needed to access the local
diff --git a/consensus/dummy/consensus.go b/consensus/dummy/consensus.go
index 89af334..e93724d 100644
--- a/consensus/dummy/consensus.go
+++ b/consensus/dummy/consensus.go
@@ -10,13 +10,13 @@ import (
        "math/big"
        "time"

-       "github.com/ava-labs/coreth/consensus"
-       "github.com/ava-labs/coreth/core/state"
-       "github.com/ava-labs/coreth/core/types"
-       "github.com/ava-labs/coreth/params"
-       "github.com/ava-labs/coreth/rpc"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/trie"
+       "gitlab.com/flarenetwork/coreth/consensus"
+       "gitlab.com/flarenetwork/coreth/core/state"
+       "gitlab.com/flarenetwork/coreth/core/types"
+       "gitlab.com/flarenetwork/coreth/params"
+       "gitlab.com/flarenetwork/coreth/rpc"
 )

 type OnFinalizeCallbackType = func(chain consensus.ChainHeaderReader, header *types.Header, state *state.StateDB, txs []*types.Transaction, receipts []*types.Receipt, uncles []*types.Header) error
diff --git a/consensus/dummy/dynamic_fees.go b/consensus/dummy/dynamic_fees.go
index 8050364..9221cec 100644
--- a/consensus/dummy/dynamic_fees.go
+++ b/consensus/dummy/dynamic_fees.go
@@ -9,10 +9,10 @@ import (
        "math/big"

        "github.com/ava-labs/avalanchego/utils/wrappers"
-       "github.com/ava-labs/coreth/core/types"
-       "github.com/ava-labs/coreth/params"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/common/math"
+       "gitlab.com/flarenetwork/coreth/core/types"
+       "gitlab.com/flarenetwork/coreth/params"
 )

 var (
diff --git a/consensus/dummy/dynamic_fees_test.go b/consensus/dummy/dynamic_fees_test.go
index fd45082..bc6813d 100644
--- a/consensus/dummy/dynamic_fees_test.go
+++ b/consensus/dummy/dynamic_fees_test.go
@@ -8,10 +8,10 @@ import (
        "math/big"
        "testing"

-       "github.com/ava-labs/coreth/core/types"
-       "github.com/ava-labs/coreth/params"
        "github.com/ethereum/go-ethereum/common/math"
        "github.com/ethereum/go-ethereum/log"
+       "gitlab.com/flarenetwork/coreth/core/types"
+       "gitlab.com/flarenetwork/coreth/params"
 )

 func testRollup(t *testing.T, longs []uint64, roll int) {
diff --git a/consensus/misc/dao.go b/consensus/misc/dao.go
index a0ab402..486f97a 100644
--- a/consensus/misc/dao.go
+++ b/consensus/misc/dao.go
@@ -31,9 +31,9 @@ import (
        "errors"
        "math/big"

-       "github.com/ava-labs/coreth/core/state"
-       "github.com/ava-labs/coreth/core/types"
-       "github.com/ava-labs/coreth/params"
```

```
+       "gitlab.com/flarenetwork/coreth/core/state"
+       "gitlab.com/flarenetwork/coreth/core/types"
+       "gitlab.com/flarenetwork/coreth/params"
 )

 var (
diff --git a/consensus/misc/forks.go b/consensus/misc/forks.go
index 6199e11..63ce4f0 100644
--- a/consensus/misc/forks.go
+++ b/consensus/misc/forks.go
@@ -29,9 +29,9 @@ package misc
 import (
        "fmt"

-       "github.com/ava-labs/coreth/core/types"
-       "github.com/ava-labs/coreth/params"
        "github.com/ethereum/go-ethereum/common"
+       "gitlab.com/flarenetwork/coreth/core/types"
+       "gitlab.com/flarenetwork/coreth/params"
 )

 // VerifyForkHashes verifies that blocks conforming to network hard-forks do have
diff --git a/core/block_validator.go b/core/block_validator.go
index f7092ee..6993bf1 100644
--- a/core/block_validator.go
+++ b/core/block_validator.go
@@ -29,11 +29,11 @@ package core
 import (
        "fmt"

-       "github.com/ava-labs/coreth/consensus"
-       "github.com/ava-labs/coreth/core/state"
-       "github.com/ava-labs/coreth/core/types"
-       "github.com/ava-labs/coreth/params"
        "github.com/ethereum/go-ethereum/trie"
+       "gitlab.com/flarenetwork/coreth/consensus"
+       "gitlab.com/flarenetwork/coreth/core/state"
+       "gitlab.com/flarenetwork/coreth/core/types"
+       "gitlab.com/flarenetwork/coreth/params"
 )

 // BlockValidator is responsible for validating block headers, uncles and
diff --git a/core/blockchain.go b/core/blockchain.go
index 7e03462..5073b6a 100644
--- a/core/blockchain.go
+++ b/core/blockchain.go
@@ -36,19 +36,19 @@ import (
        "sync/atomic"
        "time"

-       "github.com/ava-labs/coreth/consensus"
-       "github.com/ava-labs/coreth/core/rawdb"
-       "github.com/ava-labs/coreth/core/state"
-       "github.com/ava-labs/coreth/core/state/snapshot"
-       "github.com/ava-labs/coreth/core/types"
-       "github.com/ava-labs/coreth/core/vm"
-       "github.com/ava-labs/coreth/params"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/ethdb"
        "github.com/ethereum/go-ethereum/event"
        "github.com/ethereum/go-ethereum/log"
        "github.com/ethereum/go-ethereum/trie"
        lru "github.com/hashicorp/golang-lru"
+       "gitlab.com/flarenetwork/coreth/consensus"
+       "gitlab.com/flarenetwork/coreth/core/rawdb"
+       "gitlab.com/flarenetwork/coreth/core/state"
+       "gitlab.com/flarenetwork/coreth/core/state/snapshot"
+       "gitlab.com/flarenetwork/coreth/core/types"
+       "gitlab.com/flarenetwork/coreth/core/vm"
+       "gitlab.com/flarenetwork/coreth/params"
 )

 var (
diff --git a/core/blockchain_test.go b/core/blockchain_test.go
index 07cf8ec..74981e8 100644
--- a/core/blockchain_test.go
+++ b/core/blockchain_test.go
@@ -7,14 +7,14 @@ import (
        "math/big"
        "testing"

-       "github.com/ava-labs/coreth/consensus/dummy"
-       "github.com/ava-labs/coreth/core/rawdb"
-       "github.com/ava-labs/coreth/core/state"
-       "github.com/ava-labs/coreth/core/types"
-       "github.com/ava-labs/coreth/core/vm"
-       "github.com/ava-labs/coreth/params"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/ethdb"
+       "gitlab.com/flarenetwork/coreth/consensus/dummy"
+       "gitlab.com/flarenetwork/coreth/core/rawdb"
+       "gitlab.com/flarenetwork/coreth/core/state"
+       "gitlab.com/flarenetwork/coreth/core/types"
+       "gitlab.com/flarenetwork/coreth/core/vm"
+       "gitlab.com/flarenetwork/coreth/params"
 )

 func TestArchiveBlockChain(t *testing.T) {
diff --git a/core/bloom_indexer.go b/core/bloom_indexer.go
index 07e50de..c2fa061 100644
--- a/core/bloom_indexer.go
+++ b/core/bloom_indexer.go
@@ -20,12 +20,12 @@ import (
        "context"
        "time"

-       "github.com/ava-labs/coreth/core/bloombits"
-       "github.com/ava-labs/coreth/core/rawdb"
-       "github.com/ava-labs/coreth/core/types"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/common/bitutil"
        "github.com/ethereum/go-ethereum/ethdb"
+       "gitlab.com/flarenetwork/coreth/core/bloombits"
+       "gitlab.com/flarenetwork/coreth/core/rawdb"
+       "gitlab.com/flarenetwork/coreth/core/types"
 )

 const (
diff --git a/core/bloombits/generator.go b/core/bloombits/generator.go
index c0422ca..23a5b54 100644
--- a/core/bloombits/generator.go
+++ b/core/bloombits/generator.go
@@ -29,7 +29,7 @@ package bloombits
 import (
        "errors"

-       "github.com/ava-labs/coreth/core/types"
+       "gitlab.com/flarenetwork/coreth/core/types"
 )
 var (
diff --git a/core/bloombits/generator_test.go b/core/bloombits/generator_test.go
index 27dba47..b233daf 100644
--- a/core/bloombits/generator_test.go
+++ b/core/bloombits/generator_test.go
@@ -31,7 +31,7 @@ import (
        "math/rand"
        "testing"

-       "github.com/ava-labs/coreth/core/types"
+       "gitlab.com/flarenetwork/coreth/core/types"
 )

 // Tests that batched bloom bits are correctly rotated from the input bloom
diff --git a/core/chain_indexer.go b/core/chain_indexer.go
index ee6557f..eaf7881 100644
--- a/core/chain_indexer.go
+++ b/core/chain_indexer.go
@@ -34,12 +34,12 @@ import (
        "sync/atomic"
        "time"

-       "github.com/ava-labs/coreth/core/rawdb"
-       "github.com/ava-labs/coreth/core/types"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/ethdb"
        "github.com/ethereum/go-ethereum/event"
        "github.com/ethereum/go-ethereum/log"
+       "gitlab.com/flarenetwork/coreth/core/rawdb"
+       "gitlab.com/flarenetwork/coreth/core/types"
 )

 // ChainIndexerBackend defines the methods needed to process chain segments in
diff --git a/core/chain_indexer_test.go b/core/chain_indexer_test.go
index 3edf175..d8b1c43 100644
--- a/core/chain_indexer_test.go
+++ b/core/chain_indexer_test.go
@@ -35,9 +35,9 @@ import (
        "testing"
        "time"

-       "github.com/ava-labs/coreth/core/rawdb"
-       "github.com/ava-labs/coreth/core/types"
        "github.com/ethereum/go-ethereum/common"
+       "gitlab.com/flarenetwork/coreth/core/rawdb"
+       "gitlab.com/flarenetwork/coreth/core/types"
 )

 // Runs multiple tests with randomized parameters.
diff --git a/core/chain_makers.go b/core/chain_makers.go
index 4aca357..58fd71b 100644
--- a/core/chain_makers.go
+++ b/core/chain_makers.go
@@ -30,15 +30,15 @@ import (
        "fmt"
        "math/big"

-       "github.com/ava-labs/coreth/consensus"
-       "github.com/ava-labs/coreth/consensus/dummy"
-       "github.com/ava-labs/coreth/consensus/misc"
-       "github.com/ava-labs/coreth/core/state"
-       "github.com/ava-labs/coreth/core/types"
-       "github.com/ava-labs/coreth/core/vm"
-       "github.com/ava-labs/coreth/params"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/ethdb"
+       "gitlab.com/flarenetwork/coreth/consensus"
+       "gitlab.com/flarenetwork/coreth/consensus/dummy"
+       "gitlab.com/flarenetwork/coreth/consensus/misc"
+       "gitlab.com/flarenetwork/coreth/core/state"
+       "gitlab.com/flarenetwork/coreth/core/types"
+       "gitlab.com/flarenetwork/coreth/core/vm"
+       "gitlab.com/flarenetwork/coreth/params"
 )

 // BlockGen creates blocks for testing.
diff --git a/core/chain_makers_test.go b/core/chain_makers_test.go
index b67cac2..34881d4 100644
--- a/core/chain_makers_test.go
+++ b/core/chain_makers_test.go
@@ -30,13 +30,13 @@ import (
        "fmt"
        "math/big"

-       "github.com/ava-labs/coreth/consensus/dummy"
-       "github.com/ava-labs/coreth/core/rawdb"
-       "github.com/ava-labs/coreth/core/types"
-       "github.com/ava-labs/coreth/core/vm"
-       "github.com/ava-labs/coreth/params"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/crypto"
+       "gitlab.com/flarenetwork/coreth/consensus/dummy"
+       "gitlab.com/flarenetwork/coreth/core/rawdb"
+       "gitlab.com/flarenetwork/coreth/core/types"
+       "gitlab.com/flarenetwork/coreth/core/vm"
+       "gitlab.com/flarenetwork/coreth/params"
 )

 func ExampleGenerateChain() {
diff --git a/core/error.go b/core/error.go
index 0265448..4039ed7 100644
--- a/core/error.go
+++ b/core/error.go
@@ -29,7 +29,7 @@ package core
 import (
        "errors"

-       "github.com/ava-labs/coreth/core/types"
+       "gitlab.com/flarenetwork/coreth/core/types"
 )

 var (
diff --git a/core/events.go b/core/events.go
index 4898dbc..d57d378 100644
--- a/core/events.go
+++ b/core/events.go
@@ -27,8 +27,8 @@
 package core

 import (
-       "github.com/ava-labs/coreth/core/types"
        "github.com/ethereum/go-ethereum/common"
+       "gitlab.com/flarenetwork/coreth/core/types"
 )

 // NewTxsEvent is posted when a batch of transactions enter the transaction pool.
diff --git a/core/evm.go b/core/evm.go
index d45f241..2dad5a4 100644
--- a/core/evm.go
+++ b/core/evm.go
```

```
@@ -29,10 +29,10 @@ package core
 import (
        "math/big"

-       "github.com/ava-labs/coreth/consensus"
-       "github.com/ava-labs/coreth/core/types"
-       "github.com/ava-labs/coreth/core/vm"
        "github.com/ethereum/go-ethereum/common"
+       "gitlab.com/flarenetwork/coreth/consensus"
+       "gitlab.com/flarenetwork/coreth/core/types"
+       "gitlab.com/flarenetwork/coreth/core/vm"
        //"github.com/ethereum/go-ethereum/log"
 )

diff --git a/core/gen_genesis.go b/core/gen_genesis.go
index a4ec8f5..b247996 100644
--- a/core/gen_genesis.go
+++ b/core/gen_genesis.go
@@ -7,10 +7,10 @@ import (
        "errors"
        "math/big"

-       "github.com/ava-labs/coreth/params"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/common/hexutil"
        "github.com/ethereum/go-ethereum/common/math"
+       "gitlab.com/flarenetwork/coreth/params"
 )

 var _ = (*genesisSpecMarshaling)(nil)
diff --git a/core/genesis.go b/core/genesis.go
index 185887e..153f51f 100644
--- a/core/genesis.go
+++ b/core/genesis.go
@@ -34,16 +34,16 @@ import (
        "fmt"
        "math/big"

-       "github.com/ava-labs/coreth/core/rawdb"
-       "github.com/ava-labs/coreth/core/state"
-       "github.com/ava-labs/coreth/core/types"
-       "github.com/ava-labs/coreth/params"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/common/hexutil"
        "github.com/ethereum/go-ethereum/common/math"
        "github.com/ethereum/go-ethereum/ethdb"
        "github.com/ethereum/go-ethereum/log"
        "github.com/ethereum/go-ethereum/trie"
+       "gitlab.com/flarenetwork/coreth/core/rawdb"
+       "gitlab.com/flarenetwork/coreth/core/state"
+       "gitlab.com/flarenetwork/coreth/core/types"
+       "gitlab.com/flarenetwork/coreth/params"
 )

 //go:generate gencodec -type Genesis -field-override genesisSpecMarshaling -out gen_genesis.go
diff --git a/core/headerchain.go b/core/headerchain.go
index eafc747..b8d75c1 100644
--- a/core/headerchain.go
+++ b/core/headerchain.go
@@ -33,13 +33,13 @@ import (
        mrand "math/rand"
        "sync/atomic"

-       "github.com/ava-labs/coreth/consensus"
-       "github.com/ava-labs/coreth/core/rawdb"
-       "github.com/ava-labs/coreth/core/types"
-       "github.com/ava-labs/coreth/params"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/ethdb"
        lru "github.com/hashicorp/golang-lru"
+       "gitlab.com/flarenetwork/coreth/consensus"
+       "gitlab.com/flarenetwork/coreth/core/rawdb"
+       "gitlab.com/flarenetwork/coreth/core/types"
+       "gitlab.com/flarenetwork/coreth/params"
 )

 const (
diff --git a/core/keeper.go b/core/keeper.go
new file mode 100644
index 0000000..abe3acc
--- /dev/null
+++ b/core/keeper.go
@@ -0,0 +1,149 @@
+// (c) 2021, Flare Networks Limited. All rights reserved.
+// Please see the file LICENSE for licensing terms.
+
+package core
+
+import (
+       "fmt"
+       "math/big"
+
+       "github.com/ethereum/go-ethereum/common"
+       "github.com/ethereum/go-ethereum/log"
+
+       "gitlab.com/flarenetwork/coreth/core/vm"
+)
+
+// Define errors
+type ErrInvalidKeeperData struct{}
+
+func (e *ErrInvalidKeeperData) Error() string { return "invalid return data from keeper trigger" }
+
+type ErrKeeperDataEmpty struct{}
+
+func (e *ErrKeeperDataEmpty) Error() string { return "return data from keeper trigger empty" }
+
+type ErrMaxMintExceeded struct {
+       mintMax     *big.Int
+       mintRequest *big.Int
+}
+
+func (e *ErrMaxMintExceeded) Error() string {
+       return fmt.Sprintf("mint request of %s exceeded max of %s", e.mintRequest.Text(10), e.mintMax.Text(10))
+}
+
+type ErrMintNegative struct{}
+
+func (e *ErrMintNegative) Error() string { return "mint request cannot be negative" }
+
+// Define interface for dependencies
+type EVMCaller interface {
+       Call(caller vm.ContractRef, addr common.Address, input []byte, gas uint64, value *big.Int) (ret []byte, leftOverGas uint64, err error)
+       GetBlockNumber() *big.Int
+       GetGasLimit() uint64
+       AddBalance(addr common.Address, amount *big.Int)
+}
+
+// Define maximums that can change by block height
+func GetKeeperGasMultiplier(blockNumber *big.Int) uint64 {
```

```
+	switch {
+	default:
+		return 100
+	}
+}
+
+func GetSystemTriggerContractAddr(blockNumber *big.Int) string {
+	switch {
+	default:
+		return "0x1000000000000000000000000000000000000002"
+	}
+}
+
+func GetSystemTriggerSelector(blockNumber *big.Int) []byte {
+	switch {
+	default:
+		return []byte{0x7f, 0xec, 0x8d, 0x38}
+	}
+}
+
+func GetPrioritisedFTSOContract(blockTime *big.Int) string {
+	switch {
+	default:
+		return "0x1000000000000000000000000000000000000003"
+	}
+}
+
+func GetMaximumMintRequest(blockNumber *big.Int) *big.Int {
+	switch {
+	default:
+		maxRequest, _ := new(big.Int).SetString("50000000000000000000000000", 10)
+		return maxRequest
+	}
+}
+
+func triggerKeeper(evm EVMCaller) (*big.Int, error) {
+	bigZero := big.NewInt(0)
+	// Get the contract to call
+	systemTriggerContract := common.HexToAddress(GetSystemTriggerContractAddr(evm.GetBlockNumber()))
+	// Call the method
+	triggerRet, _, triggerErr := evm.Call(
+		vm.AccountRef(systemTriggerContract),
+		systemTriggerContract,
+		GetSystemTriggerSelector(evm.GetBlockNumber()),
+		GetKeeperGasMultiplier(evm.GetBlockNumber())*evm.GetGasLimit(),
+		bigZero)
+	// If no error and a value came back...
+	if triggerErr == nil && triggerRet != nil {
+		// Did we get one big int?
+		if len(triggerRet) == 32 {
+			// Convert to big int
+			// Mint request cannot be less than 0 as SetBytes treats value as unsigned
+			mintRequest := new(big.Int).SetBytes(triggerRet)
+			// return the mint request
+			return mintRequest, nil
+		} else {
+			// Returned length was not 32 bytes
+			return bigZero, &ErrInvalidKeeperData{}
+		}
+	} else {
+		if triggerErr != nil {
+			return bigZero, triggerErr
+		} else {
+			return bigZero, &ErrKeeperDataEmpty{}
+		}
+	}
+}
+
+func mint(evm EVMCaller, mintRequest *big.Int) error {
+	// If the mint request is greater than zero and less than max
+	max := GetMaximumMintRequest(evm.GetBlockNumber())
+	if mintRequest.Cmp(big.NewInt(0)) > 0 &&
+		mintRequest.Cmp(max) <= 0 {
+		// Mint the amount asked for on to the keeper contract
+		evm.AddBalance(common.HexToAddress(GetSystemTriggerContractAddr(evm.GetBlockNumber())), mintRequest)
+	} else if mintRequest.Cmp(max) > 0 {
+		// Return error
+		return &ErrMaxMintExceeded{
+			mintRequest: mintRequest,
+			mintMax:     max,
+		}
+	} else if mintRequest.Cmp(big.NewInt(0)) < 0 {
+		// Cannot mint negatives
+		return &ErrMintNegative{}
+	}
+	// No error
+	return nil
+}
+
+func triggerKeeperAndMint(evm EVMCaller, log log.Logger) {
+	// Call the keeper
+	mintRequest, triggerErr := triggerKeeper(evm)
+	// If no error...
+	if triggerErr == nil {
+		// time to mint
+		if mintError := mint(evm, mintRequest); mintError != nil {
+			log.Warn("Error minting inflation request", "error", mintError)
+		}
+	} else {
+		log.Warn("Keeper trigger in error", "error", triggerErr)
+	}
+}
diff --git a/core/keeper_test.go b/core/keeper_test.go
new file mode 100644
index 0000000..10fff7c
--- /dev/null
+++ b/core/keeper_test.go
@@ -0,0 +1,451 @@
+// (c) 2021, Flare Networks Limited. All rights reserved.
+// Please see the file LICENSE for licensing terms.
+
+package core
+
+import (
+	"errors"
+	"math/big"
+	"testing"
+
+	"github.com/ethereum/go-ethereum/common"
+	"github.com/ethereum/go-ethereum/log"
+
+	"gitlab.com/flarenetwork/coreth/core/vm"
+)
+
+// Define a mock structure to spy and mock values for keeper calls
+type MockEVMCallerData struct {
+	callCalls            int
+	addBalanceCalls      int
+	blockNumber          big.Int
+	gasLimit             uint64
```

```
+       mintRequestReturn    big.Int
+       lastAddBalanceAddr   common.Address
+       lastAddBalanceAmount *big.Int
+}
+
+// Define a mock structure to spy and mock values for logger calls
+type MockLoggerData struct {
+       warnCalls int
+}
+
+// Set up default mock method calls
+func defautCall(e *MockEVMCallerData, caller vm.ContractRef, addr common.Address, input []byte, gas uint64, value *big.Int) (ret []byte, leftOverGas uint64, err error) {
+       e.callCalls++
+
+       buffer := []byte{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
+       return e.mintRequestReturn.FillBytes(buffer), 0, nil
+}
+
+func defaultGetBlockNumber(e *MockEVMCallerData) *big.Int {
+       return &e.blockNumber
+}
+
+func defaultGetGasLimit(e *MockEVMCallerData) uint64 {
+       return e.gasLimit
+}
+
+func defaultAddBalance(e *MockEVMCallerData, addr common.Address, amount *big.Int) {
+       e.addBalanceCalls++
+       e.lastAddBalanceAddr = addr
+       e.lastAddBalanceAmount = amount
+}
+
+// Define the default EVM mock and define default mock receiver functions
+type DefaultEVMMock struct {
+       mockEVMCallerData MockEVMCallerData
+}
+
+func (e *DefaultEVMMock) Call(caller vm.ContractRef, addr common.Address, input []byte, gas uint64, value *big.Int) (ret []byte, leftOverGas uint64, err error) {
+       return defautCall(&e.mockEVMCallerData, caller, addr, input, gas, value)
+}
+
+func (e *DefaultEVMMock) GetBlockNumber() *big.Int {
+       return defaultGetBlockNumber(&e.mockEVMCallerData)
+}
+
+func (e *DefaultEVMMock) GetGasLimit() uint64 {
+       return defaultGetGasLimit(&e.mockEVMCallerData)
+}
+
+func (e *DefaultEVMMock) AddBalance(addr common.Address, amount *big.Int) {
+       defaultAddBalance(&e.mockEVMCallerData, addr, amount)
+}
+
+func TestKeeperTriggerShouldReturnMintRequest(t *testing.T) {
+       mintRequestReturn, _ := new(big.Int).SetString("50000000000000000000000000", 10)
+       mockEVMCallerData := &MockEVMCallerData{
+               blockNumber:       *big.NewInt(0),
+               gasLimit:          0,
+               mintRequestReturn: *mintRequestReturn,
+       }
+       defaultEVMMock := &DefaultEVMMock{
+               mockEVMCallerData: *mockEVMCallerData,
+       }
+
+       mintRequest, _ := triggerKeeper(defaultEVMMock)
+
+       if mintRequest.Cmp(mintRequestReturn) != 0 {
+               t.Errorf("got %s want %q", mintRequest.Text(10), "50000000000000000000000000")
+       }
+}
+
+func TestKeeperTriggerShouldNotLetMintRequestOverflow(t *testing.T) {
+       var mintRequestReturn big.Int
+       // TODO: Compact with exponent?
+       buffer := []byte{255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255}
+       mintRequestReturn.SetBytes(buffer)
+
+       mockEVMCallerData := &MockEVMCallerData{
+               blockNumber:       *big.NewInt(0),
+               gasLimit:          0,
+               mintRequestReturn: mintRequestReturn,
+       }
+       defaultEVMMock := &DefaultEVMMock{
+               mockEVMCallerData: *mockEVMCallerData,
+       }
+
+       mintRequest, mintRequestError := triggerKeeper(defaultEVMMock)
+
+       if mintRequestError != nil {
+               t.Errorf("received unexpected error %s", mintRequestError)
+       }
+
+       if mintRequest.Sign() < 1 {
+               t.Errorf("unexpected negative")
+       }
+}
+
+// Define a bad mint request return size mock
+type BadMintReturnSizeEVMMock struct {
+       mockEVMCallerData MockEVMCallerData
+}
+
+func (e *BadMintReturnSizeEVMMock) Call(caller vm.ContractRef, addr common.Address, input []byte, gas uint64, value *big.Int) (ret []byte, leftOverGas uint64, err error) {
+       e.mockEVMCallerData.callCalls++
+       // Should be size 32 bytes
+       buffer := []byte{0}
+       return e.mockEVMCallerData.mintRequestReturn.FillBytes(buffer), 0, nil
+}
+
+func (e *BadMintReturnSizeEVMMock) GetBlockNumber() *big.Int {
+       return defaultGetBlockNumber(&e.mockEVMCallerData)
+}
+
+func (e *BadMintReturnSizeEVMMock) GetGasLimit() uint64 {
+       return defaultGetGasLimit(&e.mockEVMCallerData)
+}
+
+func (e *BadMintReturnSizeEVMMock) AddBalance(addr common.Address, amount *big.Int) {
+       defaultAddBalance(&e.mockEVMCallerData, addr, amount)
+}
+
+func TestKeeperTriggerValidatesMintRequestReturnValueSize(t *testing.T) {
+       var mintRequestReturn big.Int
+       // TODO: Compact with exponent?
+       buffer := []byte{255}
+       mintRequestReturn.SetBytes(buffer)
+
+       mockEVMCallerData := &MockEVMCallerData{
+               blockNumber:       *big.NewInt(0),
+               gasLimit:          0,
+               mintRequestReturn: mintRequestReturn,
+       }
+       badMintReturnSizeEVMMock := &BadMintReturnSizeEVMMock{
+               mockEVMCallerData: *mockEVMCallerData,
+       }
+       // Call to return less than 32 bytes
+       _, err := triggerKeeper(badMintReturnSizeEVMMock)
+
+       if err != nil {
+               if err, ok := err.(*ErrInvalidKeeperData); !ok {
+                       want := &ErrInvalidKeeperData{}
+                       t.Errorf("got '%s' want '%s'", err.Error(), want.Error())
+               }
+       } else {
+               t.Errorf("no error returned as expected")
+       }
+}
+
+// Define a mock to simulate keeper trigger returning an error from Call
+type BadTriggerCallEVMMock struct {
+       mockEVMCallerData MockEVMCallerData
+}
+
+func (e *BadTriggerCallEVMMock) Call(caller vm.ContractRef, addr common.Address, input []byte, gas uint64, value *big.Int) (ret []byte, leftOverGas uint64, err error) {
+       e.mockEVMCallerData.callCalls++
+
+       buffer := []byte{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
+       return e.mockEVMCallerData.mintRequestReturn.FillBytes(buffer), 0, errors.New("Call error happened")
+}
+
+func (e *BadTriggerCallEVMMock) GetBlockNumber() *big.Int {
+       return defaultGetBlockNumber(&e.mockEVMCallerData)
+}
+
+func (e *BadTriggerCallEVMMock) GetGasLimit() uint64 {
+       return defaultGetGasLimit(&e.mockEVMCallerData)
+}
+
+func (e *BadTriggerCallEVMMock) AddBalance(addr common.Address, amount *big.Int) {
+       defaultAddBalance(&e.mockEVMCallerData, addr, amount)
+}
+
+func TestKeeperTriggerReturnsCallError(t *testing.T) {
+       mockEVMCallerData := &MockEVMCallerData{}
+       badTriggerCallEVMMock := &BadTriggerCallEVMMock{
+               mockEVMCallerData: *mockEVMCallerData,
+       }
+       // Call to return less than 32 bytes
+       _, err := triggerKeeper(badTriggerCallEVMMock)
+
+       if err == nil {
+               t.Errorf("no error received")
+       } else {
+               if err.Error() != "Call error happened" {
+                       t.Errorf("did not get expected error")
+               }
+       }
+}
+
+type LoggerMock struct {
+       mockLoggerData MockLoggerData
+}
+
+func (l *LoggerMock) New(ctx ...interface{}) log.Logger {
+       return nil
+}
+
+func (l *LoggerMock) GetHandler() log.Handler {
+       return nil
+}
+
+func (l *LoggerMock) SetHandler(h log.Handler) {
+}
+
+func (l *LoggerMock) Trace(msg string, ctx ...interface{}) {}
+func (l *LoggerMock) Debug(msg string, ctx ...interface{}) {}
+func (l *LoggerMock) Info(msg string, ctx ...interface{})  {}
+func (l *LoggerMock) Error(msg string, ctx ...interface{}) {}
+func (l *LoggerMock) Crit(msg string, ctx ...interface{})  {}
+
+func (l *LoggerMock) Warn(msg string, ctx ...interface{}) {
+       l.mockLoggerData.warnCalls++
+}
+
+func TestKeeperTriggerAndMintLogsError(t *testing.T) {
+       // Assemble
+       // Set up mock EVM call to return an error
+       mockEVMCallerData := &MockEVMCallerData{}
+       badTriggerCallEVMMock := &BadTriggerCallEVMMock{
+               mockEVMCallerData: *mockEVMCallerData,
+       }
+       // Set up a mock logger
+       mockLoggerData := &MockLoggerData{}
+       loggerMock := &LoggerMock{
+               mockLoggerData: *mockLoggerData,
+       }
+
+       // Act
+       triggerKeeperAndMint(badTriggerCallEVMMock, loggerMock)
+
+       // Assert
+       if loggerMock.mockLoggerData.warnCalls != 1 {
+               t.Errorf("Logger.Warn not called as expected")
+       }
+}
+
+// Define a mock to simulate keeper trigger returning nil for mint request
+type ReturnNilMintRequestEVMMock struct {
+       mockEVMCallerData MockEVMCallerData
+}
+
+func (e *ReturnNilMintRequestEVMMock) Call(caller vm.ContractRef, addr common.Address, input []byte, gas uint64, value *big.Int) (ret []byte, leftOverGas uint64, err error) {
+       e.mockEVMCallerData.callCalls++
+
+       return nil, 0, nil
+}
+
+func (e *ReturnNilMintRequestEVMMock) GetBlockNumber() *big.Int {
+       return defaultGetBlockNumber(&e.mockEVMCallerData)
+}
+
+func (e *ReturnNilMintRequestEVMMock) GetGasLimit() uint64 {
+       return defaultGetGasLimit(&e.mockEVMCallerData)
+}
+
+func (e *ReturnNilMintRequestEVMMock) AddBalance(addr common.Address, amount *big.Int) {
+       defaultAddBalance(&e.mockEVMCallerData, addr, amount)
+}
+
+func TestKeeperTriggerHandlesNilMintRequest(t *testing.T) {
```

```
+       mockEVMCallerData := &MockEVMCallerData{}
+       returnNilMintRequestEVMMock := &ReturnNilMintRequestEVMMock{
+               mockEVMCallerData: *mockEVMCallerData,
+       }
+       // Call to return less than 32 bytes
+       _, err := triggerKeeper(returnNilMintRequestEVMMock)
+
+       if err != nil {
+               if err, ok := err.(*ErrKeeperDataEmpty); !ok {
+                       want := &ErrKeeperDataEmpty{}
+                       t.Errorf("got '%s' want '%s'", err.Error(), want.Error())
+               }
+       } else {
+               t.Errorf("no error returned as expected")
+       }
+}
+
+func TestKeeperTriggerShouldNotMintMoreThanMax(t *testing.T) {
+       mintRequest, _ := new(big.Int).SetString("50000000000000000000000001", 10)
+       mockEVMCallerData := &MockEVMCallerData{
+               blockNumber:       *big.NewInt(0),
+               gasLimit:          0,
+               mintRequestReturn: *big.NewInt(0),
+       }
+       defaultEVMMock := &DefaultEVMMock{
+               mockEVMCallerData: *mockEVMCallerData,
+       }
+
+       err := mint(defaultEVMMock, mintRequest)
+
+       if err != nil {
+               if err, ok := err.(*ErrMaxMintExceeded); !ok {
+                       want := &ErrMaxMintExceeded{
+                               mintRequest: mintRequest,
+                               mintMax:     GetMaximumMintRequest(big.NewInt(0)),
+                       }
+                       t.Errorf("got '%s' want '%s'", err.Error(), want.Error())
+               }
+       } else {
+               t.Errorf("no error returned as expected")
+       }
+}
+
+func TestKeeperTriggerShouldNotMintNegative(t *testing.T) {
+       mintRequest := big.NewInt(-1)
+       mockEVMCallerData := &MockEVMCallerData{
+               blockNumber:       *big.NewInt(0),
+               gasLimit:          0,
+               mintRequestReturn: *big.NewInt(0),
+       }
+       defaultEVMMock := &DefaultEVMMock{
+               mockEVMCallerData: *mockEVMCallerData,
+       }
+
+       err := mint(defaultEVMMock, mintRequest)
+
+       if err != nil {
+               if err, ok := err.(*ErrMintNegative); !ok {
+                       want := &ErrMintNegative{}
+                       t.Errorf("got '%s' want '%s'", err.Error(), want.Error())
+               }
+       } else {
+               t.Errorf("no error returned as expected")
+       }
+}
+
+func TestKeeperTriggerShouldMint(t *testing.T) {
+       // Assemble
+       mintRequest, _ := new(big.Int).SetString("50000000000000000000000000", 10)
+       mockEVMCallerData := &MockEVMCallerData{
+               blockNumber:       *big.NewInt(0),
+               gasLimit:          0,
+               mintRequestReturn: *big.NewInt(0),
+       }
+       defaultEVMMock := &DefaultEVMMock{
+               mockEVMCallerData: *mockEVMCallerData,
+       }
+
+       // Act
+       err := mint(defaultEVMMock, mintRequest)
+
+       // Assert
+       if err == nil {
+               if defaultEVMMock.mockEVMCallerData.addBalanceCalls != 1 {
+                       t.Errorf("AddBalance not called as expected")
+               }
+               if defaultEVMMock.mockEVMCallerData.lastAddBalanceAddr.String() != GetSystemTriggerContractAddr(big.NewInt(0)) {
+                       t.Errorf("wanted addr %s; got addr %s", GetSystemTriggerContractAddr(big.NewInt(0)), defaultEVMMock.mockEVMCallerData.lastAddBalanceAddr)
+               }
+               if defaultEVMMock.mockEVMCallerData.lastAddBalanceAmount.Cmp(mintRequest) != 0 {
+                       t.Errorf("wanted amount %s; got amount %s", mintRequest.Text(10), defaultEVMMock.mockEVMCallerData.lastAddBalanceAmount.Text(10))
+               }
+       } else {
+               t.Errorf("unexpected error returned; was = %s", err.Error())
+       }
+}
+
+func TestKeeperTriggerShouldNotErrorMintingZero(t *testing.T) {
+       // Assemble
+       mintRequest := big.NewInt(0)
+       mockEVMCallerData := &MockEVMCallerData{
+               blockNumber:       *big.NewInt(0),
+               gasLimit:          0,
+               mintRequestReturn: *big.NewInt(0),
+       }
+       defaultEVMMock := &DefaultEVMMock{
+               mockEVMCallerData: *mockEVMCallerData,
+       }
+
+       // Act
+       err := mint(defaultEVMMock, mintRequest)
+
+       // Assert
+       if err == nil {
+               if defaultEVMMock.mockEVMCallerData.addBalanceCalls != 0 {
+                       t.Errorf("AddBalance called unexpectedly")
+               }
+       } else {
+               t.Errorf("unexpected error returned; was %s", err.Error())
+       }
+}
+
+func TestKeeperTriggerFiredAndMinted(t *testing.T) {
+       mintRequestReturn, _ := new(big.Int).SetString("50000000000000000000000000", 10)
+       mockEVMCallerData := &MockEVMCallerData{
+               blockNumber:       *big.NewInt(0),
+               gasLimit:          0,
+               mintRequestReturn: *mintRequestReturn,
+       }
+       defaultEVMMock := &DefaultEVMMock{
```

```
+		mockEVMCallerData: *mockEVMCallerData,
+	}
+
+	log := log.New()
+	triggerKeeperAndMint(defaultEVMMock, log)
+
+	// EVM Call function calling the keeper should have been cqlled
+	if defaultEVMMock.mockEVMCallerData.callCalls != 1 {
+		t.Errorf("EVM Call count not as expected. got %d want 1", defaultEVMMock.mockEVMCallerData.callCalls)
+	}
+	// AddBalance should have been called on the state database, minting the request asked for
+	if defaultEVMMock.mockEVMCallerData.addBalanceCalls != 1 {
+		t.Errorf("Add balance call count not as expected. got %d want 1", defaultEVMMock.mockEVMCallerData.addBalanceCalls)
+	}
+}
+
+func TestKeeperTriggerShouldNotMintMoreThanLimit(t *testing.T) {
+	mintRequestReturn, _ := new(big.Int).SetString("50000000000000000000000001", 10)
+	mockEVMCallerData := &MockEVMCallerData{
+		blockNumber:       *big.NewInt(0),
+		gasLimit:          0,
+		mintRequestReturn: *mintRequestReturn,
+	}
+	defaultEVMMock := &DefaultEVMMock{
+		mockEVMCallerData: *mockEVMCallerData,
+	}
+
+	log := log.New()
+	triggerKeeperAndMint(defaultEVMMock, log)
+
+	// EVM Call function calling the keeper should have been called
+	if defaultEVMMock.mockEVMCallerData.callCalls != 1 {
+		t.Errorf("EVM Call count not as expected. got %d want 1", defaultEVMMock.mockEVMCallerData.callCalls)
+	}
+	// AddBalance should not have been called on the state database, as the mint request was over the limit
+	if defaultEVMMock.mockEVMCallerData.addBalanceCalls != 0 {
+		t.Errorf("Add balance call count not as expected. got %d want 1", defaultEVMMock.mockEVMCallerData.addBalanceCalls)
+	}
+}
diff --git a/core/mkalloc.go b/core/mkalloc.go
index 96f9c31..046924b 100644
--- a/core/mkalloc.go
+++ b/core/mkalloc.go
@@ -24,6 +24,7 @@
 // You should have received a copy of the GNU Lesser General Public License
 // along with the go-ethereum library. If not, see <http://www.gnu.org/licenses/>.
 
+//go:build none
 // +build none
 
 /*
@@ -44,8 +45,8 @@ import (
 	"sort"
 	"strconv"
 
-	"github.com/ava-labs/coreth/core"
 	"github.com/ethereum/go-ethereum/rlp"
+	"gitlab.com/flarenetwork/coreth/core"
 )
 
 type allocItem struct{ Addr, Balance *big.Int }
diff --git a/core/rawdb/accessors_chain.go b/core/rawdb/accessors_chain.go
index 6ed9e0a..13e0f1a 100644
--- a/core/rawdb/accessors_chain.go
+++ b/core/rawdb/accessors_chain.go
@@ -31,13 +31,13 @@ import (
 	"encoding/binary"
 	"math/big"
 
-	"github.com/ava-labs/coreth/core/types"
-	"github.com/ava-labs/coreth/params"
 	"github.com/ethereum/go-ethereum/common"
 	"github.com/ethereum/go-ethereum/crypto"
 	"github.com/ethereum/go-ethereum/ethdb"
 	"github.com/ethereum/go-ethereum/log"
 	"github.com/ethereum/go-ethereum/rlp"
+	"gitlab.com/flarenetwork/coreth/core/types"
+	"gitlab.com/flarenetwork/coreth/params"
 )
 
 // ReadCanonicalHash retrieves the hash assigned to a canonical block number.
diff --git a/core/rawdb/accessors_chain_test.go b/core/rawdb/accessors_chain_test.go
index 38372a1..55b3ab4 100644
--- a/core/rawdb/accessors_chain_test.go
+++ b/core/rawdb/accessors_chain_test.go
@@ -24,10 +24,10 @@ import (
 	"reflect"
 	"testing"
 
-	"github.com/ava-labs/coreth/core/types"
-	"github.com/ava-labs/coreth/params"
 	"github.com/ethereum/go-ethereum/common"
 	"github.com/ethereum/go-ethereum/rlp"
+	"gitlab.com/flarenetwork/coreth/core/types"
+	"gitlab.com/flarenetwork/coreth/params"
 	"golang.org/x/crypto/sha3"
 )
 
diff --git a/core/rawdb/accessors_indexes.go b/core/rawdb/accessors_indexes.go
index 3c8b467..4cec301 100644
--- a/core/rawdb/accessors_indexes.go
+++ b/core/rawdb/accessors_indexes.go
@@ -30,12 +30,12 @@ import (
 	"bytes"
 	"math/big"
 
-	"github.com/ava-labs/coreth/core/types"
-	"github.com/ava-labs/coreth/params"
 	"github.com/ethereum/go-ethereum/common"
 	"github.com/ethereum/go-ethereum/ethdb"
 	"github.com/ethereum/go-ethereum/log"
 	"github.com/ethereum/go-ethereum/rlp"
+	"gitlab.com/flarenetwork/coreth/core/types"
+	"gitlab.com/flarenetwork/coreth/params"
 )
 
 // ReadTxLookupEntry retrieves the positional metadata associated with a transaction
diff --git a/core/rawdb/accessors_indexes_test.go b/core/rawdb/accessors_indexes_test.go
index a80ff1e..8efdc25 100644
--- a/core/rawdb/accessors_indexes_test.go
+++ b/core/rawdb/accessors_indexes_test.go
@@ -22,10 +22,10 @@ import (
 	"math/big"
 	"testing"
 
-	"github.com/ava-labs/coreth/core/types"
 	"github.com/ethereum/go-ethereum/common"
 	"github.com/ethereum/go-ethereum/ethdb"
 	"github.com/ethereum/go-ethereum/rlp"
+	"gitlab.com/flarenetwork/coreth/core/types"
```

```
	"golang.org/x/crypto/sha3"
 )

diff --git a/core/rawdb/accessors_metadata.go b/core/rawdb/accessors_metadata.go
index 5b39253..47b9e9d 100644
--- a/core/rawdb/accessors_metadata.go
+++ b/core/rawdb/accessors_metadata.go
@@ -29,11 +29,11 @@ package rawdb
 import (
 	"encoding/json"
 
-	"github.com/ava-labs/coreth/params"
 	"github.com/ethereum/go-ethereum/common"
 	"github.com/ethereum/go-ethereum/ethdb"
 	"github.com/ethereum/go-ethereum/log"
 	"github.com/ethereum/go-ethereum/rlp"
+	"gitlab.com/flarenetwork/coreth/params"
 )
 
 // ReadDatabaseVersion retrieves the version number of the database.
diff --git a/core/state/database.go b/core/state/database.go
index f4af1a0..feba572 100644
--- a/core/state/database.go
+++ b/core/state/database.go
@@ -31,11 +31,11 @@ import (
 	"fmt"
 
 	"github.com/VictoriaMetrics/fastcache"
-	"github.com/ava-labs/coreth/core/rawdb"
 	"github.com/ethereum/go-ethereum/common"
 	"github.com/ethereum/go-ethereum/ethdb"
 	"github.com/ethereum/go-ethereum/trie"
 	lru "github.com/hashicorp/golang-lru"
+	"gitlab.com/flarenetwork/coreth/core/rawdb"
 )
 
 const (
diff --git a/core/state/snapshot/conversion.go b/core/state/snapshot/conversion.go
index 0fd4e9b..d2b519f 100644
--- a/core/state/snapshot/conversion.go
+++ b/core/state/snapshot/conversion.go
@@ -36,12 +36,12 @@ import (
 	"sync"
 	"time"
 
-	"github.com/ava-labs/coreth/core/rawdb"
 	"github.com/ethereum/go-ethereum/common"
 	"github.com/ethereum/go-ethereum/ethdb"
 	"github.com/ethereum/go-ethereum/log"
 	"github.com/ethereum/go-ethereum/rlp"
 	"github.com/ethereum/go-ethereum/trie"
+	"gitlab.com/flarenetwork/coreth/core/rawdb"
 )
 
 // trieKV represents a trie key-value pair
diff --git a/core/state/snapshot/disklayer.go b/core/state/snapshot/disklayer.go
index 106c451..c33f4a6 100644
--- a/core/state/snapshot/disklayer.go
+++ b/core/state/snapshot/disklayer.go
@@ -32,11 +32,11 @@ import (
 	"time"
 
 	"github.com/VictoriaMetrics/fastcache"
-	"github.com/ava-labs/coreth/core/rawdb"
 	"github.com/ethereum/go-ethereum/common"
 	"github.com/ethereum/go-ethereum/ethdb"
 	"github.com/ethereum/go-ethereum/rlp"
 	"github.com/ethereum/go-ethereum/trie"
+	"gitlab.com/flarenetwork/coreth/core/rawdb"
 )
 
 // diskLayer is a low level persistent snapshot built on top of a key-value store.
diff --git a/core/state/snapshot/disklayer_test.go b/core/state/snapshot/disklayer_test.go
index f423c9a..9d758be 100644
--- a/core/state/snapshot/disklayer_test.go
+++ b/core/state/snapshot/disklayer_test.go
@@ -32,12 +32,12 @@ import (
 	"os"
 	"testing"
 
-	"github.com/ava-labs/coreth/core/rawdb"
 	"github.com/ethereum/go-ethereum/common"
 	"github.com/ethereum/go-ethereum/ethdb"
 	"github.com/ethereum/go-ethereum/ethdb/leveldb"
 	"github.com/ethereum/go-ethereum/ethdb/memorydb"
 	"github.com/ethereum/go-ethereum/rlp"
+	"gitlab.com/flarenetwork/coreth/core/rawdb"
 )
 
 // reverse reverses the contents of a byte slice. It's used to update random accs
diff --git a/core/state/snapshot/generate.go b/core/state/snapshot/generate.go
index 45f927f..3a48d1f 100644
--- a/core/state/snapshot/generate.go
+++ b/core/state/snapshot/generate.go
@@ -34,7 +34,6 @@ import (
 	"time"
 
 	"github.com/VictoriaMetrics/fastcache"
-	"github.com/ava-labs/coreth/core/rawdb"
 	"github.com/ethereum/go-ethereum/common"
 	"github.com/ethereum/go-ethereum/common/math"
 	"github.com/ethereum/go-ethereum/crypto"
@@ -42,6 +41,7 @@ import (
 	"github.com/ethereum/go-ethereum/log"
 	"github.com/ethereum/go-ethereum/rlp"
 	"github.com/ethereum/go-ethereum/trie"
+	"gitlab.com/flarenetwork/coreth/core/rawdb"
 )
 
 var (
diff --git a/core/state/snapshot/iterator.go b/core/state/snapshot/iterator.go
index 7a5da6a..2c34768 100644
--- a/core/state/snapshot/iterator.go
+++ b/core/state/snapshot/iterator.go
@@ -31,9 +31,9 @@ import (
 	"fmt"
 	"sort"
 
-	"github.com/ava-labs/coreth/core/rawdb"
 	"github.com/ethereum/go-ethereum/common"
 	"github.com/ethereum/go-ethereum/ethdb"
+	"gitlab.com/flarenetwork/coreth/core/rawdb"
 )
 
 // Iterator is an iterator to step over all the accounts or the specific
diff --git a/core/state/snapshot/iterator_test.go b/core/state/snapshot/iterator_test.go
index 9d1b744..1916c49 100644
--- a/core/state/snapshot/iterator_test.go
+++ b/core/state/snapshot/iterator_test.go
@@ -33,8 +33,8 @@ import (
```

```diff
 	"math/rand"
 	"testing"
 
-	"github.com/ava-labs/coreth/core/rawdb"
 	"github.com/ethereum/go-ethereum/common"
+	"gitlab.com/flarenetwork/coreth/core/rawdb"
 )
 
 // TestAccountIteratorBasics tests some simple single-layer(diff and disk) iteration
diff --git a/core/state/snapshot/journal.go b/core/state/snapshot/journal.go
index 145f467..96c7c2e 100644
--- a/core/state/snapshot/journal.go
+++ b/core/state/snapshot/journal.go
@@ -33,12 +33,12 @@ import (
 	"time"
 
 	"github.com/VictoriaMetrics/fastcache"
-	"github.com/ava-labs/coreth/core/rawdb"
 	"github.com/ethereum/go-ethereum/common"
 	"github.com/ethereum/go-ethereum/ethdb"
 	"github.com/ethereum/go-ethereum/log"
 	"github.com/ethereum/go-ethereum/rlp"
 	"github.com/ethereum/go-ethereum/trie"
+	"gitlab.com/flarenetwork/coreth/core/rawdb"
 )
 
 // journalGenerator is a disk layer entry containing the generator progress marker.
diff --git a/core/state/snapshot/snapshot.go b/core/state/snapshot/snapshot.go
index bbdba6b..e311d69 100644
--- a/core/state/snapshot/snapshot.go
+++ b/core/state/snapshot/snapshot.go
@@ -36,12 +36,12 @@ import (
 	"time"
 
 	"github.com/VictoriaMetrics/fastcache"
-	"github.com/ava-labs/coreth/core/rawdb"
 	"github.com/ethereum/go-ethereum/common"
 	"github.com/ethereum/go-ethereum/ethdb"
 	"github.com/ethereum/go-ethereum/log"
 	"github.com/ethereum/go-ethereum/metrics"
 	"github.com/ethereum/go-ethereum/trie"
+	"gitlab.com/flarenetwork/coreth/core/rawdb"
 )
 
 const (
diff --git a/core/state/snapshot/snapshot_test.go b/core/state/snapshot/snapshot_test.go
index c28dafa..da5cd63 100644
--- a/core/state/snapshot/snapshot_test.go
+++ b/core/state/snapshot/snapshot_test.go
@@ -32,9 +32,9 @@ import (
 	"math/rand"
 	"testing"
 
-	"github.com/ava-labs/coreth/core/rawdb"
 	"github.com/ethereum/go-ethereum/common"
 	"github.com/ethereum/go-ethereum/rlp"
+	"gitlab.com/flarenetwork/coreth/core/rawdb"
 )
 
 // randomHash generates a random blob of data and returns it as a hash.
diff --git a/core/state/snapshot/wipe.go b/core/state/snapshot/wipe.go
index ca1f7e0..3cbf754 100644
--- a/core/state/snapshot/wipe.go
+++ b/core/state/snapshot/wipe.go
@@ -30,10 +30,10 @@ import (
 	"bytes"
 	"time"
 
-	"github.com/ava-labs/coreth/core/rawdb"
 	"github.com/ethereum/go-ethereum/common"
 	"github.com/ethereum/go-ethereum/ethdb"
 	"github.com/ethereum/go-ethereum/log"
+	"gitlab.com/flarenetwork/coreth/core/rawdb"
 )
 
 // wipeSnapshot starts a goroutine to iterate over the entire key-value database
diff --git a/core/state/snapshot/wipe_test.go b/core/state/snapshot/wipe_test.go
index 952370d..ba4ce69 100644
--- a/core/state/snapshot/wipe_test.go
+++ b/core/state/snapshot/wipe_test.go
@@ -30,9 +30,9 @@ import (
 	"math/rand"
 	"testing"
 
-	"github.com/ava-labs/coreth/core/rawdb"
 	"github.com/ethereum/go-ethereum/common"
 	"github.com/ethereum/go-ethereum/ethdb/memorydb"
+	"gitlab.com/flarenetwork/coreth/core/rawdb"
 )
 
 // Tests that given a database with random data content, all parts of a snapshot
diff --git a/core/state/state_test.go b/core/state/state_test.go
index 849f661..ec66ccc 100644
--- a/core/state/state_test.go
+++ b/core/state/state_test.go
@@ -27,9 +27,9 @@
 package state
 
 import (
-	"github.com/ava-labs/coreth/core/rawdb"
 	"github.com/ethereum/go-ethereum/common"
 	"github.com/ethereum/go-ethereum/ethdb"
+	"gitlab.com/flarenetwork/coreth/core/rawdb"
 )
 
 type stateTest struct {
diff --git a/core/state/statedb.go b/core/state/statedb.go
index acc779c..3fe062c 100644
--- a/core/state/statedb.go
+++ b/core/state/statedb.go
@@ -34,15 +34,15 @@ import (
 	"sort"
 	"time"
 
-	"github.com/ava-labs/coreth/core/rawdb"
-	"github.com/ava-labs/coreth/core/state/snapshot"
-	"github.com/ava-labs/coreth/core/types"
 	"github.com/ethereum/go-ethereum/common"
 	"github.com/ethereum/go-ethereum/crypto"
 	"github.com/ethereum/go-ethereum/log"
 	"github.com/ethereum/go-ethereum/metrics"
 	"github.com/ethereum/go-ethereum/rlp"
 	"github.com/ethereum/go-ethereum/trie"
+	"gitlab.com/flarenetwork/coreth/core/rawdb"
+	"gitlab.com/flarenetwork/coreth/core/state/snapshot"
+	"gitlab.com/flarenetwork/coreth/core/types"
 )
 
 type revision struct {
diff --git a/core/state/statedb_test.go b/core/state/statedb_test.go
```

```
index 9c295f9..4841b6b 100644
--- a/core/state/statedb_test.go
+++ b/core/state/statedb_test.go
@@ -39,11 +39,11 @@ import (
        "testing"
        "testing/quick"
 
-       "github.com/ava-labs/coreth/core/rawdb"
-       "github.com/ava-labs/coreth/core/state/snapshot"
-       "github.com/ava-labs/coreth/core/types"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/crypto"
+       "gitlab.com/flarenetwork/coreth/core/rawdb"
+       "gitlab.com/flarenetwork/coreth/core/state/snapshot"
+       "gitlab.com/flarenetwork/coreth/core/types"
 )
 
 // Tests that updating a state trie does not leak any database writes prior to
diff --git a/core/state_connector.go b/core/state_connector.go
new file mode 100644
index 0000000..db6665e
--- /dev/null
+++ b/core/state_connector.go
@@ -0,0 +1,731 @@
+// (c) 2021, Flare Networks Limited. All rights reserved.
+// Please see the file LICENSE for licensing terms.
+
+package core
+
+import (
+       "bytes"
+       "crypto/sha256"
+       "encoding/binary"
+       "encoding/hex"
+       "encoding/json"
+       "fmt"
+       "io/ioutil"
+       "math"
+       "math/big"
+       "net/http"
+       "os"
+       "strconv"
+       "strings"
+       "time"
+
+       "github.com/ethereum/go-ethereum/common"
+       "github.com/ethereum/go-ethereum/common/hexutil"
+       "github.com/ethereum/go-ethereum/crypto"
+)
+
+var (
+       testingChainID               = new(big.Int).SetUint64(16)
+       stateConnectorActivationTime = new(big.Int).SetUint64(1636070400)
+       tr                           = &http.Transport{
+               MaxIdleConns:        100,
+               MaxConnsPerHost:     100,
+               MaxIdleConnsPerHost: 100,
+               IdleConnTimeout:     60 * time.Second,
+               DisableCompression:  true,
+       }
+       client = &http.Client{
+               Transport: tr,
+               Timeout:   5 * time.Second,
+       }
+       apiRetries    = 3
+       apiRetryDelay = 1 * time.Second
+)
+
+func GetStateConnectorActivated(chainID *big.Int, blockTime *big.Int) bool {
+       // Return true if chainID is 16 or if block.timestamp is greater than the state connector activation time on any chain
+       return chainID.Cmp(testingChainID) == 0 || blockTime.Cmp(stateConnectorActivationTime) > 0
+}
+
+func GetStateConnectorGasDivisor(blockTime *big.Int) uint64 {
+       switch {
+       default:
+               return 3
+       }
+}
+
+func GetMaxAllowedChains(blockTime *big.Int) uint32 {
+       switch {
+       default:
+               return 5
+       }
+}
+
+func GetStateConnectorContractAddr(blockTime *big.Int) string {
+       switch {
+       default:
+               return "0x1000000000000000000000000000000000000001"
+       }
+}
+
+func GetProveDataAvailabilityPeriodFinalitySelector(blockTime *big.Int) []byte {
+       switch {
+       default:
+               return []byte{0xc5, 0xd6, 0x4c, 0xd1}
+       }
+}
+
+func GetProvePaymentFinalitySelector(blockTime *big.Int) []byte {
+       switch {
+       default:
+               return []byte{0x38, 0x84, 0x92, 0xdd}
+       }
+}
+
+func GetDisprovePaymentFinalitySelector(blockTime *big.Int) []byte {
+       switch {
+       default:
+               return []byte{0x7f, 0x58, 0x24, 0x32}
+       }
+}
+
+// =======================================================
+// Proof of Work Common
+// =======================================================
+
+type GetPoWRequestPayload struct {
+       Method string   `json:"method"`
+       Params []string `json:"params"`
+}
+type GetPoWBlockCountResp struct {
+       Result uint64      `json:"result"`
+       Error  interface{} `json:"error"`
+}
+
+func GetPoWBlockCount(chainURL string, username string, password string) (uint64, bool) {
+       data := GetPoWRequestPayload{
```

```
+               Method: "getblockcount",
+               Params: []string{},
+       }
+       payloadBytes, err := json.Marshal(data)
+       if err != nil {
+               return 0, true
+       }
+       body := bytes.NewReader(payloadBytes)
+       req, err := http.NewRequest("POST", chainURL, body)
+       if err != nil {
+               return 0, true
+       }
+       req.Header.Set("Content-Type", "application/json")
+       if username != "" && password != "" {
+               req.SetBasicAuth(username, password)
+       }
+       resp, err := client.Do(req)
+       if err != nil {
+               return 0, true
+       }
+       defer resp.Body.Close()
+       if resp.StatusCode != 200 {
+               return 0, true
+       }
+       respBody, err := ioutil.ReadAll(resp.Body)
+       if err != nil {
+               return 0, true
+       }
+       var jsonResp GetPoWBlockCountResp
+       err = json.Unmarshal(respBody, &jsonResp)
+       if err != nil {
+               return 0, true
+       }
+       if jsonResp.Error != nil {
+               return 0, true
+       }
+       return jsonResp.Result, false
+}
+
+type GetPoWBlockHeaderResult struct {
+       Hash          string `json:"hash"`
+       Confirmations uint64 `json:"confirmations"`
+       Height        uint64 `json:"height"`
+}
+type GetPoWBlockHeaderResp struct {
+       Result GetPoWBlockHeaderResult `json:"result"`
+       Error  interface{}             `json:"error"`
+}
+
+func GetPoWBlockHeader(ledgerHash string, requiredConfirmations uint64, chainURL string, username string, password string) (uint64, bool) {
+       data := GetPoWRequestPayload{
+               Method: "getblockheader",
+               Params: []string{
+                       ledgerHash,
+               },
+       }
+       payloadBytes, err := json.Marshal(data)
+       if err != nil {
+               return 0, true
+       }
+       body := bytes.NewReader(payloadBytes)
+       req, err := http.NewRequest("POST", chainURL, body)
+       if err != nil {
+               return 0, true
+       }
+       req.Header.Set("Content-Type", "application/json")
+       if username != "" && password != "" {
+               req.SetBasicAuth(username, password)
+       }
+       resp, err := client.Do(req)
+       if err != nil {
+               return 0, true
+       }
+       defer resp.Body.Close()
+       if resp.StatusCode != 200 {
+               return 0, true
+       }
+       respBody, err := ioutil.ReadAll(resp.Body)
+       if err != nil {
+               return 0, true
+       }
+       var jsonResp GetPoWBlockHeaderResp
+       err = json.Unmarshal(respBody, &jsonResp)
+       if err != nil {
+               return 0, true
+       }
+       if jsonResp.Error != nil {
+               return 0, false
+       } else if jsonResp.Result.Confirmations < requiredConfirmations {
+               return 0, false
+       }
+       return jsonResp.Result.Height, false
+}
+
+func ProveDataAvailabilityPeriodFinalityPoW(checkRet []byte, chainURL string, username string, password string) (bool, bool) {
+       blockCount, err := GetPoWBlockCount(chainURL, username, password)
+       if err {
+               return false, true
+       }
+       ledger := binary.BigEndian.Uint64(checkRet[56:64])
+       requiredConfirmations := binary.BigEndian.Uint64(checkRet[88:96])
+       if blockCount < ledger+requiredConfirmations {
+               return false, true
+       }
+       ledgerResp, err := GetPoWBlockHeader(hex.EncodeToString(checkRet[96:128]), requiredConfirmations, chainURL, username, password)
+       if err {
+               return false, true
+       } else if ledgerResp > 0 && ledgerResp == ledger {
+               return true, false
+       } else {
+               return false, false
+       }
+}
+
+type GetPoWTxRequestParams struct {
+       TxID    string `json:"txid"`
+       Verbose bool   `json:"verbose"`
+}
+type GetPoWTxRequestPayload struct {
+       Method string                `json:"method"`
+       Params GetPoWTxRequestParams `json:"params"`
+}
+type GetPoWTxResult struct {
+       TxID          string `json:"txid"`
+       BlockHash     string `json:"blockhash"`
+       Confirmations uint64 `json:"confirmations"`
+       Vout          []struct {
+               Value        float64 `json:"value"`
+               N            uint64  `json:"n"`
+               ScriptPubKey struct {
```

```
+                       Type      string   `json:"type"`
+                       Addresses []string `json:"addresses"`
+               } `json:"scriptPubKey"`
+       } `json:"vout"`
+}
+type GetPoWTxResp struct {
+       Result GetPoWTxResult `json:"result"`
+       Error  interface{}    `json:"error"`
+}
+
+func GetPoWTx(txHash string, voutN uint64, latestAvailableBlock uint64, currencyCode string, chainURL string, username string, password string) ([]byte, uint64, bool) {
+       data := GetPoWTxRequestPayload{
+               Method: "getrawtransaction",
+               Params: GetPoWTxRequestParams{
+                       TxID:    txHash[1:],
+                       Verbose: true,
+               },
+       }
+       payloadBytes, err := json.Marshal(data)
+       if err != nil {
+               return []byte{}, 0, true
+       }
+       body := bytes.NewReader(payloadBytes)
+       req, err := http.NewRequest("POST", chainURL, body)
+       if err != nil {
+               return []byte{}, 0, true
+       }
+       req.Header.Set("Content-Type", "application/json")
+       if username != "" && password != "" {
+               req.SetBasicAuth(username, password)
+       }
+       resp, err := client.Do(req)
+       if err != nil {
+               return []byte{}, 0, true
+       }
+       defer resp.Body.Close()
+       if resp.StatusCode != 200 {
+               return []byte{}, 0, true
+       }
+       respBody, err := ioutil.ReadAll(resp.Body)
+       if err != nil {
+               return []byte{}, 0, true
+       }
+       var jsonResp GetPoWTxResp
+       err = json.Unmarshal(respBody, &jsonResp)
+       if err != nil {
+               return []byte{}, 0, true
+       }
+       if jsonResp.Error != nil {
+               return []byte{}, 0, true
+       }
+       if uint64(len(jsonResp.Result.Vout)) <= voutN {
+               return []byte{}, 0, false
+       }
+       if jsonResp.Result.Vout[voutN].ScriptPubKey.Type != "pubkeyhash" || len(jsonResp.Result.Vout[voutN].ScriptPubKey.Addresses) != 1 {
+               return []byte{}, 0, false
+       }
+       inBlock, getBlockErr := GetPoWBlockHeader(jsonResp.Result.BlockHash, jsonResp.Result.Confirmations, chainURL, username, password)
+       if getBlockErr {
+               return []byte{}, 0, true
+       }
+       if inBlock == 0 || inBlock >= latestAvailableBlock {
+               return []byte{}, 0, false
+       }
+       txIdHash := crypto.Keccak256([]byte(txHash))
+       destinationHash := crypto.Keccak256([]byte(jsonResp.Result.Vout[voutN].ScriptPubKey.Addresses[0]))
+       amountHash := crypto.Keccak256(common.LeftPadBytes(common.FromHex(hexutil.EncodeUint64(uint64(jsonResp.Result.Vout[voutN].Value*math.Pow(10, 8)))), 32))
+       currencyHash := crypto.Keccak256([]byte(currencyCode))
+       return crypto.Keccak256(txIdHash, destinationHash, amountHash, currencyHash), inBlock, false
+}
+
+func ProvePaymentFinalityPoW(checkRet []byte, isDisprove bool, currencyCode string, chainURL string, username string, password string) (bool, bool) {
+       if len(checkRet) < 257 {
+               return false, false
+       }
+       voutN, err := strconv.ParseUint(string(checkRet[192:193]), 16, 64)
+       if err != nil {
+               return false, false
+       }
+       paymentHash, inBlock, getPoWTxErr := GetPoWTx(string(checkRet[192:257]), voutN, binary.BigEndian.Uint64(checkRet[88:96]), currencyCode, chainURL, username, password)
+       if getPoWTxErr {
+               return false, true
+       }
+       if !isDisprove {
+               if len(paymentHash) > 0 && bytes.Equal(paymentHash, checkRet[96:128]) && inBlock == binary.BigEndian.Uint64(checkRet[56:64]) {
+                       return true, false
+               }
+       } else {
+               if len(paymentHash) > 0 && bytes.Equal(paymentHash, checkRet[96:128]) && inBlock > binary.BigEndian.Uint64(checkRet[56:64]) {
+                       return true, false
+               } else if len(paymentHash) == 0 {
+                       return true, false
+               }
+       }
+       return false, false
+}
+
+func ProvePoW(sender common.Address, blockTime *big.Int, functionSelector []byte, checkRet []byte, currencyCode string, chainURL string) (bool, bool) {
+       var username, password string
+       chainURLhash := sha256.Sum256([]byte(chainURL))
+       chainURLchecksum := hex.EncodeToString(chainURLhash[0:4])
+       switch currencyCode {
+       case "btc":
+               username = os.Getenv("BTC_U_" + chainURLchecksum)
+               password = os.Getenv("BTC_P_" + chainURLchecksum)
+       case "ltc":
+               username = os.Getenv("LTC_U_" + chainURLchecksum)
+               password = os.Getenv("LTC_P_" + chainURLchecksum)
+       case "dog":
+               username = os.Getenv("DOGE_U_" + chainURLchecksum)
+               password = os.Getenv("DOGE_P_" + chainURLchecksum)
+       }
+       if bytes.Equal(functionSelector, GetProveDataAvailabilityPeriodFinalitySelector(blockTime)) {
+               return ProveDataAvailabilityPeriodFinalityPoW(checkRet, chainURL, username, password)
+       } else if bytes.Equal(functionSelector, GetProvePaymentFinalitySelector(blockTime)) {
+               return ProvePaymentFinalityPoW(checkRet, false, currencyCode, chainURL, username, password)
+       } else if bytes.Equal(functionSelector, GetDisprovePaymentFinalitySelector(blockTime)) {
+               return ProvePaymentFinalityPoW(checkRet, true, currencyCode, chainURL, username, password)
+       }
+       return false, false
+}
+
+// =======================================================
+// XRP
+// =======================================================
+
+type GetXRPBlockRequestParams struct {
+       LedgerIndex  uint64 `json:"ledger_index"`
+       Full         bool   `json:"full"`
+       Accounts     bool   `json:"accounts"`
+       Transactions bool   `json:"transactions"`
+       Expand       bool   `json:"expand"`
+       OwnerFunds   bool   `json:"owner_funds"`
+}
+type GetXRPBlockRequestPayload struct {
+       Method string                      `json:"method"`
+       Params []GetXRPBlockRequestParams `json:"params"`
+}
+type CheckXRPErrorResponse struct {
+       Error string `json:"error"`
+}
+type GetXRPBlockResponse struct {
+       LedgerHash  string `json:"ledger_hash"`
+       LedgerIndex int    `json:"ledger_index"`
+       Validated   bool   `json:"validated"`
+}
+
+func GetXRPBlock(ledger uint64, chainURL string) (string, bool) {
+       data := GetXRPBlockRequestPayload{
+               Method: "ledger",
+               Params: []GetXRPBlockRequestParams{
+                       {
+                               LedgerIndex:  ledger,
+                               Full:         false,
+                               Accounts:     false,
+                               Transactions: false,
+                               Expand:       false,
+                               OwnerFunds:   false,
+                       },
+               },
+       }
+       payloadBytes, err := json.Marshal(data)
+       if err != nil {
+               return "", true
+       }
+       body := bytes.NewReader(payloadBytes)
+       req, err := http.NewRequest("POST", chainURL, body)
+       if err != nil {
+               return "", true
+       }
+       req.Header.Set("Content-Type", "application/json")
+       resp, err := client.Do(req)
+       if err != nil {
+               return "", true
+       }
+       defer resp.Body.Close()
+       if resp.StatusCode != 200 {
+               return "", true
+       }
+       respBody, err := ioutil.ReadAll(resp.Body)
+       if err != nil {
+               return "", true
+       }
+       var checkErrorResp map[string]CheckXRPErrorResponse
+       err = json.Unmarshal(respBody, &checkErrorResp)
+       if err != nil {
+               return "", true
+       }
+       if checkErrorResp["result"].Error != "" {
+               return "", true
+       }
+       var jsonResp map[string]GetXRPBlockResponse
+       err = json.Unmarshal(respBody, &jsonResp)
+       if err != nil {
+               return "", true
+       }
+       if !jsonResp["result"].Validated {
+               return "", true
+       }
+       return jsonResp["result"].LedgerHash, false
+}
+
+func ProveDataAvailabilityPeriodFinalityXRP(checkRet []byte, chainURL string) (bool, bool) {
+       ledger := binary.BigEndian.Uint64(checkRet[56:64])
+       ledgerHashString, err := GetXRPBlock(ledger, chainURL)
+       if err {
+               return false, true
+       }
+       if ledgerHashString != "" && bytes.Equal(crypto.Keccak256([]byte(ledgerHashString)), checkRet[96:128]) {
+               return true, false
+       }
+       return false, false
+}
+
+type GetXRPTxRequestParams struct {
+       Transaction string `json:"transaction"`
+       Binary      bool   `json:"binary"`
+}
+type GetXRPTxRequestPayload struct {
+       Method string                   `json:"method"`
+       Params []GetXRPTxRequestParams `json:"params"`
+}
+type GetXRPTxResponse struct {
+       Destination     string `json:"Destination"`
+       DestinationTag  int    `json:"DestinationTag"`
+       TransactionType string `json:"TransactionType"`
+       Hash            string `json:"hash"`
+       InLedger        int    `json:"inLedger"`
+       Validated       bool   `json:"validated"`
+       Meta            struct {
+               TransactionResult string      `json:"TransactionResult"`
+               Amount            interface{} `json:"delivered_amount"`
+       } `json:"meta"`
+}
+
+type GetXRPTxIssuedCurrency struct {
+       Currency string `json:"currency"`
+       Issuer   string `json:"issuer"`
+       Value    string `json:"value"`
+}
+
+func GetXRPTx(txHash string, latestAvailableLedger uint64, chainURL string) ([]byte, uint64, bool) {
+       data := GetXRPTxRequestPayload{
+               Method: "tx",
+               Params: []GetXRPTxRequestParams{
+                       {
+                               Transaction: txHash,
+                               Binary:      false,
+                       },
+               },
+       }
+       payloadBytes, err := json.Marshal(data)
+       if err != nil {
+               return []byte{}, 0, true
+       }
+       body := bytes.NewReader(payloadBytes)
+       req, err := http.NewRequest("POST", chainURL, body)
+       if err != nil {
+               return []byte{}, 0, true
+       }
+       req.Header.Set("Content-Type", "application/json")
+       resp, err := client.Do(req)
+       if err != nil {
+               return []byte{}, 0, true
+       }
+       defer resp.Body.Close()
+       if resp.StatusCode != 200 {
+               return []byte{}, 0, true
+       }
+       respBody, err := ioutil.ReadAll(resp.Body)
+       if err != nil {
+               return []byte{}, 0, true
+       }
+       var checkErrorResp map[string]CheckXRPErrorResponse
+       err = json.Unmarshal(respBody, &checkErrorResp)
+       if err != nil {
+               return []byte{}, 0, true
+       }
+       respErrString := checkErrorResp["result"].Error
+       if respErrString != "" {
+               if respErrString == "amendmentBlocked" ||
+                       respErrString == "failedToForward" ||
+                       respErrString == "invalid_API_version" ||
+                       respErrString == "noClosed" ||
+                       respErrString == "noCurrent" ||
+                       respErrString == "noNetwork" ||
+                       respErrString == "tooBusy" {
+                       return []byte{}, 0, true
+               } else {
+                       return []byte{}, 0, false
+               }
+       }
+       var jsonResp map[string]GetXRPTxResponse
+       err = json.Unmarshal(respBody, &jsonResp)
+       if err != nil {
+               return []byte{}, 0, false
+       }
+       if jsonResp["result"].TransactionType != "Payment" || !jsonResp["result"].Validated || jsonResp["result"].Meta.TransactionResult != "tesSUCCESS" {
+               return []byte{}, 0, false
+       }
+       inLedger := uint64(jsonResp["result"].InLedger)
+       if inLedger == 0 || inLedger >= latestAvailableLedger || !jsonResp["result"].Validated {
+               return []byte{}, 0, false
+       }
+       var currency string
+       var amount uint64
+       if stringAmount, ok := jsonResp["result"].Meta.Amount.(string); ok {
+               amount, err = strconv.ParseUint(stringAmount, 10, 64)
+               if err != nil {
+                       return []byte{}, 0, false
+               }
+               currency = "xrp"
+       } else {
+               amountStruct, err := json.Marshal(jsonResp["result"].Meta.Amount)
+               if err != nil {
+                       return []byte{}, 0, false
+               }
+               var issuedCurrencyResp GetXRPTxIssuedCurrency
+               err = json.Unmarshal(amountStruct, &issuedCurrencyResp)
+               if err != nil {
+                       return []byte{}, 0, false
+               }
+               floatAmount, err := strconv.ParseFloat(issuedCurrencyResp.Value, 64)
+               if err != nil {
+                       return []byte{}, 0, false
+               }
+               amount = uint64(floatAmount * math.Pow(10, 15))
+               currency = issuedCurrencyResp.Currency + issuedCurrencyResp.Issuer
+       }
+       txIdHash := crypto.Keccak256([]byte(jsonResp["result"].Hash))
+       destinationHash := crypto.Keccak256([]byte(jsonResp["result"].Destination))
+       destinationTagHash := crypto.Keccak256(common.LeftPadBytes(common.FromHex(hexutil.EncodeUint64(uint64(jsonResp["result"].DestinationTag))), 32))
+       destinationHash = crypto.Keccak256(destinationHash, destinationTagHash)
+       amountHash := crypto.Keccak256(common.LeftPadBytes(common.FromHex(hexutil.EncodeUint64(amount)), 32))
+       currencyHash := crypto.Keccak256([]byte(currency))
+       return crypto.Keccak256(txIdHash, destinationHash, amountHash, currencyHash), inLedger, false
+}
+
+func ProvePaymentFinalityXRP(checkRet []byte, isDisprove bool, chainURL string) (bool, bool) {
+       paymentHash, inLedger, err := GetXRPTx(string(checkRet[192:]), binary.BigEndian.Uint64(checkRet[88:96]), chainURL)
+       if err {
+               return false, true
+       }
+       if !isDisprove {
+               if len(paymentHash) > 0 && bytes.Equal(paymentHash, checkRet[96:128]) && inLedger == binary.BigEndian.Uint64(checkRet[56:64]) {
+                       return true, false
+               }
+       } else {
+               if len(paymentHash) > 0 && bytes.Equal(paymentHash, checkRet[96:128]) && inLedger > binary.BigEndian.Uint64(checkRet[56:64]) {
+                       return true, false
+               } else if len(paymentHash) == 0 {
+                       return true, false
+               }
+       }
+       return false, false
+}
+
+func ProveXRP(sender common.Address, blockTime *big.Int, functionSelector []byte, checkRet []byte, chainURL string) (bool, bool) {
+       if bytes.Equal(functionSelector, GetProveDataAvailabilityPeriodFinalitySelector(blockTime)) {
+               return ProveDataAvailabilityPeriodFinalityXRP(checkRet, chainURL)
+       } else if bytes.Equal(functionSelector, GetProvePaymentFinalitySelector(blockTime)) {
+               return ProvePaymentFinalityXRP(checkRet, false, chainURL)
+       } else if bytes.Equal(functionSelector, GetDisprovePaymentFinalitySelector(blockTime)) {
+               return ProvePaymentFinalityXRP(checkRet, true, chainURL)
+       }
+       return false, false
+}
+
+// =======================================================
+// ALGO
+// =======================================================
+
+func ProveALGO(sender common.Address, blockTime *big.Int, functionSelector []byte, checkRet []byte, chainURL string) (bool, bool) {
+       return false, false
+}
+
+// =======================================================
+// Common
+// =======================================================
+
+func ProveChain(sender common.Address, blockTime *big.Int, functionSelector []byte, checkRet []byte, chainId uint32, chainURL string) (bool, bool) {
+       switch chainId {
+       case 0:
+               return ProvePoW(sender, blockTime, functionSelector, checkRet, "btc", chainURL)
+       case 1:
+               return ProvePoW(sender, blockTime, functionSelector, checkRet, "ltc", chainURL)
+       case 2:
+               return ProvePoW(sender, blockTime, functionSelector, checkRet, "dog", chainURL)
+       case 3:
+               return ProveXRP(sender, blockTime, functionSelector, checkRet, chainURL)
+       case 4:
+               return ProveALGO(sender, blockTime, functionSelector, checkRet, chainURL)
+       default:
+               return false, true
+       }
+}
+
+func ReadChain(sender common.Address, blockTime *big.Int, functionSelector []byte, checkRet []byte) bool {
+       chainId := binary.BigEndian.Uint32(checkRet[28:32])
+       var chainURLs string
+       switch chainId {
+       case 0:
+               chainURLs = os.Getenv("BTC_APIs")
+       case 1:
+               chainURLs = os.Getenv("LTC_APIs")
+       case 2:
+               chainURLs = os.Getenv("DOGE_APIs")
+       case 3:
+               chainURLs = os.Getenv("XRP_APIs")
+       case 4:
+               chainURLs = os.Getenv("ALGO_APIs")
+       }
+       if chainURLs == "" {
+               return false
+       }
+       for i := 0; i < apiRetries; i++ {
+               for _, chainURL := range strings.Split(chainURLs, ",") {
+                       if chainURL == "" {
+                               continue
+                       }
+                       verified, err := ProveChain(sender, blockTime, functionSelector, checkRet, chainId, chainURL)
+                       if !verified && err {
+                               continue
+                       }
+                       return verified
+               }
+               time.Sleep(apiRetryDelay)
+       }
+       return false
+}
+
+func GetVerificationPaths(functionSelector []byte, checkRet []byte) (string, string) {
+       prefix := "cache/"
+       acceptedPrefix := "ACCEPTED"
+       rejectedPrefix := "REJECTED"
+       functionHash := hex.EncodeToString(functionSelector[:])
+       verificationHash := hex.EncodeToString(crypto.Keccak256(checkRet[0:64], checkRet[96:128]))
+       suffix := "_" + functionHash + "_" + verificationHash
+       return prefix + acceptedPrefix + suffix, prefix + rejectedPrefix + suffix
+}
+
+// Verify proof against underlying chain
+func StateConnectorCall(sender common.Address, blockTime *big.Int, functionSelector []byte, checkRet []byte) bool {
+       if binary.BigEndian.Uint64(checkRet[88:96]) > 0 {
+               go func() {
+                       acceptedPath, rejectedPath := GetVerificationPaths(functionSelector, checkRet)
+                       _, errACCEPTED := os.Stat(acceptedPath)
+                       _, errREJECTED := os.Stat(rejectedPath)
+                       if errACCEPTED != nil && errREJECTED != nil {
+                               if ReadChain(sender, blockTime, functionSelector, checkRet) {
+                                       verificationHashStore, err := os.Create(acceptedPath)
+                                       verificationHashStore.Close()
+                                       if err != nil {
+                                               // Permissions problem
+                                               panic(err)
+                                       }
+                               } else {
+                                       verificationHashStore, err := os.Create(rejectedPath)
+                                       verificationHashStore.Close()
+                                       if err != nil {
+                                               // Permissions problem
+                                               panic(err)
+                                       }
+                               }
+                       }
+               }()
+               return true
+       } else {
+               acceptedPath, rejectedPath := GetVerificationPaths(functionSelector, checkRet)
+               _, errACCEPTED := os.Stat(acceptedPath)
+               _, errREJECTED := os.Stat(rejectedPath)
+               if errACCEPTED != nil && errREJECTED != nil {
+                       for i := 0; i < 2*apiRetries; i++ {
+                               _, errACCEPTED = os.Stat(acceptedPath)
+                               _, errREJECTED = os.Stat(rejectedPath)
+                               if errACCEPTED == nil || errREJECTED == nil {
+                                       break
+                               }
+                               time.Sleep(apiRetryDelay)
+                       }
+               }
+               go func() {
+                       removeFulfilledAPIRequests := os.Getenv("REMOVE_FULFILLED_API_REQUESTS")
+                       if removeFulfilledAPIRequests == "1" {
+                               errDeleteACCEPTED := os.Remove(acceptedPath)
+                               errDeleteREJECTED := os.Remove(rejectedPath)
+                               if errDeleteACCEPTED != nil && errDeleteREJECTED != nil {
+                                       // Permissions problem
+                                       panic(fmt.Sprintf("%s\n%s", errDeleteACCEPTED, errDeleteREJECTED))
+                               }
+                       }
+               }()
+               return errACCEPTED == nil
+       }
+}
diff --git a/core/state_manager.go b/core/state_manager.go
index 8b76069..c95e03f 100644
--- a/core/state_manager.go
+++ b/core/state_manager.go
@@ -30,9 +30,9 @@ import (
        "fmt"
        "math/rand"

-       "github.com/ava-labs/coreth/core/types"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/ethdb"
+       "gitlab.com/flarenetwork/coreth/core/types"
 )

 const (
diff --git a/core/state_manager_test.go b/core/state_manager_test.go
index 17e2b1d..f57d105 100644
--- a/core/state_manager_test.go
+++ b/core/state_manager_test.go
@@ -7,7 +7,7 @@ import (
        "math/big"
        "testing"

-       "github.com/ava-labs/coreth/core/types"
+       "gitlab.com/flarenetwork/coreth/core/types"
```

```
        "github.com/ethereum/go-ethereum/common"
        "github.com/stretchr/testify/assert"
diff --git a/core/state_prefetcher.go b/core/state_prefetcher.go
index a1e0cde..f0fd973 100644
--- a/core/state_prefetcher.go
+++ b/core/state_prefetcher.go
@@ -30,11 +30,11 @@ import (
        "math/big"
        "sync/atomic"

-       "github.com/ava-labs/coreth/consensus"
-       "github.com/ava-labs/coreth/core/state"
-       "github.com/ava-labs/coreth/core/types"
-       "github.com/ava-labs/coreth/core/vm"
-       "github.com/ava-labs/coreth/params"
+       "gitlab.com/flarenetwork/coreth/consensus"
+       "gitlab.com/flarenetwork/coreth/core/state"
+       "gitlab.com/flarenetwork/coreth/core/types"
+       "gitlab.com/flarenetwork/coreth/core/vm"
+       "gitlab.com/flarenetwork/coreth/params"
 )

 // statePrefetcher is a basic Prefetcher, which blindly executes a block on top
diff --git a/core/state_processor.go b/core/state_processor.go
index b9c243f..ec6d709 100644
--- a/core/state_processor.go
+++ b/core/state_processor.go
@@ -30,14 +30,14 @@ import (
        "fmt"
        "math/big"

-       "github.com/ava-labs/coreth/consensus"
-       "github.com/ava-labs/coreth/consensus/misc"
-       "github.com/ava-labs/coreth/core/state"
-       "github.com/ava-labs/coreth/core/types"
-       "github.com/ava-labs/coreth/core/vm"
-       "github.com/ava-labs/coreth/params"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/crypto"
+       "gitlab.com/flarenetwork/coreth/consensus"
+       "gitlab.com/flarenetwork/coreth/consensus/misc"
+       "gitlab.com/flarenetwork/coreth/core/state"
+       "gitlab.com/flarenetwork/coreth/core/types"
+       "gitlab.com/flarenetwork/coreth/core/vm"
+       "gitlab.com/flarenetwork/coreth/params"
 )

 // StateProcessor is a basic Processor, which takes care of transitioning
diff --git a/core/state_transition.go b/core/state_transition.go
index 381327c..876d39d 100644
--- a/core/state_transition.go
+++ b/core/state_transition.go
@@ -1,3 +1,11 @@
+// (c) 2021, Flare Networks Limited. All rights reserved.
+//
+// This file is a derived work, based on the avalanchego library whose original
+// notices appear below. It is distributed under a license compatible with the
+// licensing terms of the original code from which it is derived.
+// Please see the file LICENSE_AVALABS for licensing terms of the original work.
+// Please see the file LICENSE for licensing terms.
+//
 // (c) 2019-2020, Ava Labs, Inc.
 //
 // This file is a derived work, based on the go-ethereum library whose original
@@ -27,16 +35,19 @@
 package core

 import (
+       "bytes"
+       "encoding/binary"
        "fmt"
        "math"
        "math/big"

+       "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/crypto"
+       "github.com/ethereum/go-ethereum/log"

-       "github.com/ava-labs/coreth/core/types"
-       "github.com/ava-labs/coreth/core/vm"
-       "github.com/ava-labs/coreth/params"
-       "github.com/ethereum/go-ethereum/common"
+       "gitlab.com/flarenetwork/coreth/core/types"
+       "gitlab.com/flarenetwork/coreth/core/vm"
+       "gitlab.com/flarenetwork/coreth/params"
 )

 var emptyCodeHash = crypto.Keccak256Hash(nil)
@@ -115,6 +126,23 @@ func (result *ExecutionResult) Return() []byte {
        return common.CopyBytes(result.ReturnData)
 }

+// Implement the EVMCaller interface on the state transition structure; simply delegate the calls
+func (st *StateTransition) Call(caller vm.ContractRef, addr common.Address, input []byte, gas uint64, value *big.Int) (ret []byte, leftOverGas uint64, err error) {
+       return st.evm.Call(caller, addr, input, gas, value)
+}
+
+func (st *StateTransition) GetBlockNumber() *big.Int {
+       return st.evm.Context.BlockNumber
+}
+
+func (st *StateTransition) GetGasLimit() uint64 {
+       return st.evm.Context.GasLimit
+}
+
+func (st *StateTransition) AddBalance(addr common.Address, amount *big.Int) {
+       st.state.AddBalance(addr, amount)
+}
+
 // Revert returns the concrete revert reason if the execution is aborted by `REVERT`
 // opcode. Note the reason can be nil if no data supplied with revert opcode.
 func (result *ExecutionResult) Revert() []byte {
@@ -321,19 +349,91 @@ func (st *StateTransition) TransitionDb() (*ExecutionResult, error) {
        if rules := st.evm.ChainConfig().AvalancheRules(st.evm.Context.BlockNumber, st.evm.Context.Time); rules.IsApricotPhase2 {
                st.state.PrepareAccessList(msg.From(), msg.To(), vm.ActivePrecompiles(rules), msg.AccessList())
        }
+
        var (
-               ret   []byte
-               vmerr error // vm errors do not effect consensus and are therefore not assigned to err
+               ret                                      []byte
+               vmerr                                    error // vm errors do not affect consensus and are therefore not assigned to err
+               selectProveDataAvailabilityPeriodFinality bool
+               selectProvePaymentFinality               bool
+               selectDisprovePaymentFinality            bool
+               prioritisedFTSOContract                  bool
        )
-       if contractCreation {
-               ret, _, st.gas, vmerr = st.evm.Create(sender, st.data, st.gas, st.value)
```

```diff
-	} else {
+
+	if st.evm.Context.Coinbase != common.HexToAddress("0x0100000000000000000000000000000000000000") {
+		return nil, fmt.Errorf("Invalid value for block.coinbase")
+	}
+	if st.msg.From() == common.HexToAddress("0x0100000000000000000000000000000000000000") ||
+		st.msg.From() == common.HexToAddress(GetStateConnectorContractAddr(st.evm.Context.Time)) ||
+		st.msg.From() == common.HexToAddress(GetSystemTriggerContractAddr(st.evm.Context.Time)) {
+		return nil, fmt.Errorf("Invalid sender")
+	}
+	burnAddress := st.evm.Context.Coinbase
+	if !contractCreation {
+		if *msg.To() == common.HexToAddress(GetStateConnectorContractAddr(st.evm.Context.Time)) && len(st.data) >= 4 {
+			selectProveDataAvailabilityPeriodFinality = bytes.Equal(st.data[0:4], GetProveDataAvailabilityPeriodFinalitySelector(st.evm.Context.Time))
+			selectProvePaymentFinality = bytes.Equal(st.data[0:4], GetProvePaymentFinalitySelector(st.evm.Context.Time))
+			selectDisprovePaymentFinality = bytes.Equal(st.data[0:4], GetDisprovePaymentFinalitySelector(st.evm.Context.Time))
+		} else {
+			prioritisedFTSOContract = *msg.To() == common.HexToAddress(GetPrioritisedFTSOContract(st.evm.Context.Time))
+		}
+	}
+
+	if selectProveDataAvailabilityPeriodFinality || selectProvePaymentFinality || selectDisprovePaymentFinality {
 		// Increment the nonce for the next transaction
 		st.state.SetNonce(msg.From(), st.state.GetNonce(sender.Address())+1)
-		ret, st.gas, vmerr = st.evm.Call(sender, st.to(), st.data, st.gas, st.value)
+		stateConnectorGas := st.gas / GetStateConnectorGasDivisor(st.evm.Context.Time)
+		checkRet, _, checkVmerr := st.evm.Call(sender, st.to(), st.data, stateConnectorGas, st.value)
+		if checkVmerr == nil {
+			chainConfig := st.evm.ChainConfig()
+			if GetStateConnectorActivated(chainConfig.ChainID, st.evm.Context.Time) && binary.BigEndian.Uint32(checkRet[28:32]) < GetMaxAllowedChains(st.evm.Context.Time) {
+				if StateConnectorCall(msg.From(), st.evm.Context.Time, st.data[0:4], checkRet) {
+					originalCoinbase := st.evm.Context.Coinbase
+					defer func() {
+						st.evm.Context.Coinbase = originalCoinbase
+					}()
+					st.evm.Context.Coinbase = st.msg.From()
+				}
+			}
+		}
+		ret, st.gas, vmerr = st.evm.Call(sender, st.to(), st.data, stateConnectorGas, st.value)
+	} else {
+		if contractCreation {
+			ret, _, st.gas, vmerr = st.evm.Create(sender, st.data, st.gas, st.value)
+		} else {
+			// Increment the nonce for the next transaction
+			st.state.SetNonce(msg.From(), st.state.GetNonce(sender.Address())+1)
+			ret, st.gas, vmerr = st.evm.Call(sender, st.to(), st.data, st.gas, st.value)
+		}
 	}
 	st.refundGas(apricotPhase1)
-	st.state.AddBalance(st.evm.Context.Coinbase, new(big.Int).Mul(new(big.Int).SetUint64(st.gasUsed()), st.gasPrice))
+	if vmerr == nil && prioritisedFTSOContract {
+		nominalGasUsed := uint64(21000)
+		nominalGasPrice := uint64(225_000_000_000)
+		nominalFee := new(big.Int).Mul(new(big.Int).SetUint64(nominalGasUsed), new(big.Int).SetUint64(nominalGasPrice))
+		actualGasUsed := st.gasUsed()
+		actualGasPrice := st.gasPrice
+		actualFee := new(big.Int).Mul(new(big.Int).SetUint64(actualGasUsed), actualGasPrice)
+		if actualFee.Cmp(nominalFee) > 0 {
+			feeRefund := new(big.Int).Sub(actualFee, nominalFee)
+			st.state.AddBalance(st.msg.From(), feeRefund)
+			st.state.AddBalance(burnAddress, nominalFee)
+		} else {
+			st.state.AddBalance(burnAddress, actualFee)
+		}
+	} else {
+		st.state.AddBalance(burnAddress, new(big.Int).Mul(new(big.Int).SetUint64(st.gasUsed()), st.gasPrice))
+	}
+
+	// Call the keeper contract trigger method if there is no vm error
+	if vmerr == nil {
+		// Temporarily disable EVM debugging
+		oldDebug := st.evm.Config.Debug
+		st.evm.Config.Debug = false
+		// Call the keeper contract trigger
+		log := log.Root()
+		triggerKeeperAndMint(st, log)
+		st.evm.Config.Debug = oldDebug
+	}
 
 	return &ExecutionResult{
 		UsedGas:    st.gasUsed(),
diff --git a/core/test_blockchain.go b/core/test_blockchain.go
index 604ed0f..cfdd2b9 100644
--- a/core/test_blockchain.go
+++ b/core/test_blockchain.go
@@ -8,13 +8,13 @@ import (
 	"math/big"
 	"testing"
 
-	"github.com/ava-labs/coreth/core/rawdb"
-	"github.com/ava-labs/coreth/core/state"
-	"github.com/ava-labs/coreth/core/types"
-	"github.com/ava-labs/coreth/params"
 	"github.com/ethereum/go-ethereum/common"
 	"github.com/ethereum/go-ethereum/crypto"
 	"github.com/ethereum/go-ethereum/ethdb"
+	"gitlab.com/flarenetwork/coreth/core/rawdb"
+	"gitlab.com/flarenetwork/coreth/core/state"
+	"gitlab.com/flarenetwork/coreth/core/types"
+	"gitlab.com/flarenetwork/coreth/params"
 )
 
 type ChainTest struct {
diff --git a/core/tx_cacher.go b/core/tx_cacher.go
index 86679ff..03099f2 100644
--- a/core/tx_cacher.go
+++ b/core/tx_cacher.go
@@ -29,7 +29,7 @@ package core
 import (
 	"runtime"
 
-	"github.com/ava-labs/coreth/core/types"
+	"gitlab.com/flarenetwork/coreth/core/types"
 )
 
 // senderCacher is a concurrent transaction sender recoverer and cacher.
diff --git a/core/tx_journal.go b/core/tx_journal.go
index b2bfa53..17b4419 100644
--- a/core/tx_journal.go
+++ b/core/tx_journal.go
@@ -31,10 +31,10 @@ import (
 	"io"
 	"os"
 
-	"github.com/ava-labs/coreth/core/types"
 	"github.com/ethereum/go-ethereum/common"
 	"github.com/ethereum/go-ethereum/log"
 	"github.com/ethereum/go-ethereum/rlp"
+	"gitlab.com/flarenetwork/coreth/core/types"
```

```
 )

 // errNoActiveJournal is returned if a transaction is attempted to be inserted
diff --git a/core/tx_list.go b/core/tx_list.go
index 6788bad..2e28237 100644
--- a/core/tx_list.go
+++ b/core/tx_list.go
@@ -33,8 +33,8 @@ import (
        "sort"
        "time"

-       "github.com/ava-labs/coreth/core/types"
        "github.com/ethereum/go-ethereum/common"
+       "gitlab.com/flarenetwork/coreth/core/types"
 )

 // nonceHeap is a heap.Interface implementation over 64bit unsigned integers for
diff --git a/core/tx_list_test.go b/core/tx_list_test.go
index 86cf79a..91e4d6e 100644
--- a/core/tx_list_test.go
+++ b/core/tx_list_test.go
@@ -31,8 +31,8 @@ import (
        "math/rand"
        "testing"

-       "github.com/ava-labs/coreth/core/types"
        "github.com/ethereum/go-ethereum/crypto"
+       "gitlab.com/flarenetwork/coreth/core/types"
 )

 // Tests that transactions can be added to strict lists and list contents and
diff --git a/core/tx_noncer.go b/core/tx_noncer.go
index bb5968b..439b9c7 100644
--- a/core/tx_noncer.go
+++ b/core/tx_noncer.go
@@ -29,8 +29,8 @@ package core
 import (
        "sync"

-       "github.com/ava-labs/coreth/core/state"
        "github.com/ethereum/go-ethereum/common"
+       "gitlab.com/flarenetwork/coreth/core/state"
 )

 // txNoncer is a tiny virtual state database to manage the executable nonces of
diff --git a/core/tx_pool.go b/core/tx_pool.go
index 457be8f..9663b17 100644
--- a/core/tx_pool.go
+++ b/core/tx_pool.go
@@ -35,15 +35,15 @@ import (
        "sync"
        "time"

-       "github.com/ava-labs/coreth/consensus/dummy"
-       "github.com/ava-labs/coreth/core/state"
-       "github.com/ava-labs/coreth/core/types"
-       "github.com/ava-labs/coreth/params"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/common/prque"
        "github.com/ethereum/go-ethereum/event"
        "github.com/ethereum/go-ethereum/log"
        "github.com/ethereum/go-ethereum/metrics"
+       "gitlab.com/flarenetwork/coreth/consensus/dummy"
+       "gitlab.com/flarenetwork/coreth/core/state"
+       "gitlab.com/flarenetwork/coreth/core/types"
+       "gitlab.com/flarenetwork/coreth/params"
 )

 const (
diff --git a/core/tx_pool_test.go b/core/tx_pool_test.go
index 84353e9..97e6c7e 100644
--- a/core/tx_pool_test.go
+++ b/core/tx_pool_test.go
@@ -29,14 +29,14 @@ import (
        "testing"
        "time"

-       "github.com/ava-labs/coreth/core/rawdb"
-       "github.com/ava-labs/coreth/core/state"
-       "github.com/ava-labs/coreth/core/types"
-       "github.com/ava-labs/coreth/params"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/crypto"
        "github.com/ethereum/go-ethereum/event"
        "github.com/ethereum/go-ethereum/trie"
+       "gitlab.com/flarenetwork/coreth/core/rawdb"
+       "gitlab.com/flarenetwork/coreth/core/state"
+       "gitlab.com/flarenetwork/coreth/core/types"
+       "gitlab.com/flarenetwork/coreth/params"
 )

 var (
diff --git a/core/types.go b/core/types.go
index 9607212..6a0d938 100644
--- a/core/types.go
+++ b/core/types.go
@@ -27,9 +27,9 @@
 package core

 import (
-       "github.com/ava-labs/coreth/core/state"
-       "github.com/ava-labs/coreth/core/types"
-       "github.com/ava-labs/coreth/core/vm"
+       "gitlab.com/flarenetwork/coreth/core/state"
+       "gitlab.com/flarenetwork/coreth/core/types"
+       "gitlab.com/flarenetwork/coreth/core/vm"
 )

 // Validator is an interface which defines the standard for block validation. It
diff --git a/core/types/block_test.go b/core/types/block_test.go
index e2f5623..85db341 100644
--- a/core/types/block_test.go
+++ b/core/types/block_test.go
@@ -33,11 +33,11 @@ import (
        "reflect"
        "testing"

-       "github.com/ava-labs/coreth/params"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/common/math"
        "github.com/ethereum/go-ethereum/crypto"
        "github.com/ethereum/go-ethereum/rlp"
+       "gitlab.com/flarenetwork/coreth/params"
        "golang.org/x/crypto/sha3"
 )

diff --git a/core/types/hashing_test.go b/core/types/hashing_test.go
index 424798d..e537e47 100644
--- a/core/types/hashing_test.go
+++ b/core/types/hashing_test.go
```

```
@@ -34,12 +34,12 @@ import (
        mrand "math/rand"
        "testing"

-       "github.com/ava-labs/coreth/core/types"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/common/hexutil"
        "github.com/ethereum/go-ethereum/crypto"
        "github.com/ethereum/go-ethereum/rlp"
        "github.com/ethereum/go-ethereum/trie"
+       "gitlab.com/flarenetwork/coreth/core/types"
 )

 func TestDeriveSha(t *testing.T) {
diff --git a/core/types/receipt.go b/core/types/receipt.go
index 6da7c15..90366de 100644
--- a/core/types/receipt.go
+++ b/core/types/receipt.go
@@ -34,11 +34,11 @@ import (
        "math/big"
        "unsafe"

-       "github.com/ava-labs/coreth/params"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/common/hexutil"
        "github.com/ethereum/go-ethereum/crypto"
        "github.com/ethereum/go-ethereum/rlp"
+       "gitlab.com/flarenetwork/coreth/params"
 )

 //go:generate gencodec -type Receipt -field-override receiptMarshaling -out gen_receipt_json.go
diff --git a/core/types/receipt_test.go b/core/types/receipt_test.go
index 95fe68a..d2a2508 100644
--- a/core/types/receipt_test.go
+++ b/core/types/receipt_test.go
@@ -33,10 +33,10 @@ import (
        "reflect"
        "testing"

-       "github.com/ava-labs/coreth/params"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/crypto"
        "github.com/ethereum/go-ethereum/rlp"
+       "gitlab.com/flarenetwork/coreth/params"
 )

 func TestDecodeEmptyTypedReceipt(t *testing.T) {
diff --git a/core/types/transaction_signing.go b/core/types/transaction_signing.go
index a717749..982d10d 100644
--- a/core/types/transaction_signing.go
+++ b/core/types/transaction_signing.go
@@ -32,9 +32,9 @@ import (
        "fmt"
        "math/big"

-       "github.com/ava-labs/coreth/params"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/crypto"
+       "gitlab.com/flarenetwork/coreth/params"
 )

 var ErrInvalidChainId = errors.New("invalid chain id for signer")
diff --git a/core/vm/access_list_tracer.go b/core/vm/access_list_tracer.go
index c481c54..22c8f3e 100644
--- a/core/vm/access_list_tracer.go
+++ b/core/vm/access_list_tracer.go
@@ -30,8 +30,8 @@ import (
        "math/big"
        "time"

-       "github.com/ava-labs/coreth/core/types"
        "github.com/ethereum/go-ethereum/common"
+       "gitlab.com/flarenetwork/coreth/core/types"
 )

 // accessList is an accumulator for the set of accounts and storage slots an EVM
diff --git a/core/vm/contracts.go b/core/vm/contracts.go
index cfb2671..31b8d8f 100644
--- a/core/vm/contracts.go
+++ b/core/vm/contracts.go
@@ -32,13 +32,13 @@ import (
        "errors"
        "math/big"

-       "github.com/ava-labs/coreth/params"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/common/math"
        "github.com/ethereum/go-ethereum/crypto"
        "github.com/ethereum/go-ethereum/crypto/blake2b"
        "github.com/ethereum/go-ethereum/crypto/bls12381"
        "github.com/ethereum/go-ethereum/crypto/bn256"
+       "gitlab.com/flarenetwork/coreth/params"

        //lint:ignore SA1019 Needed for precompile
        "golang.org/x/crypto/ripemd160"
diff --git a/core/vm/contracts_stateful.go b/core/vm/contracts_stateful.go
index 981ebd7..05b0361 100644
--- a/core/vm/contracts_stateful.go
+++ b/core/vm/contracts_stateful.go
@@ -7,9 +7,9 @@ import (
        "fmt"
        "math/big"

-       "github.com/ava-labs/coreth/params"
        "github.com/ethereum/go-ethereum/common"
        "github.com/holiman/uint256"
+       "gitlab.com/flarenetwork/coreth/params"
 )

 // PrecompiledContractsApricot contains the default set of pre-compiled Ethereum
diff --git a/core/vm/contracts_stateful_test.go b/core/vm/contracts_stateful_test.go
index 6126078..1278a67 100644
--- a/core/vm/contracts_stateful_test.go
+++ b/core/vm/contracts_stateful_test.go
@@ -7,12 +7,12 @@ import (
        "math/big"
        "testing"

-       "github.com/ava-labs/coreth/core/rawdb"
-       "github.com/ava-labs/coreth/core/state"
-       "github.com/ava-labs/coreth/params"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/log"
        "github.com/stretchr/testify/assert"
+       "gitlab.com/flarenetwork/coreth/core/rawdb"
+       "gitlab.com/flarenetwork/coreth/core/state"
+       "gitlab.com/flarenetwork/coreth/params"
 )

 func TestPrecompiledContractSpendsGas(t *testing.T) {
```

```
diff --git a/core/vm/eips.go b/core/vm/eips.go
index e525a73..399a874 100644
--- a/core/vm/eips.go
+++ b/core/vm/eips.go
@@ -30,8 +30,8 @@ import (
        "fmt"
        "sort"

-       "github.com/ava-labs/coreth/params"
        "github.com/holiman/uint256"
+       "gitlab.com/flarenetwork/coreth/params"
 )

 var activators = map[int]func(*JumpTable){
diff --git a/core/vm/evm.go b/core/vm/evm.go
index 9ad9db1..6f78cc6 100644
--- a/core/vm/evm.go
+++ b/core/vm/evm.go
@@ -31,10 +31,10 @@ import (
        "sync/atomic"
        "time"

-       "github.com/ava-labs/coreth/params"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/crypto"
        "github.com/holiman/uint256"
+       "gitlab.com/flarenetwork/coreth/params"
 )

 // emptyCodeHash is used by create to ensure deployment is disallowed to already
diff --git a/core/vm/gas_table.go b/core/vm/gas_table.go
index 383fb6d..d611fc5 100644
--- a/core/vm/gas_table.go
+++ b/core/vm/gas_table.go
@@ -29,9 +29,9 @@ package vm
 import (
        "errors"

-       "github.com/ava-labs/coreth/params"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/common/math"
+       "gitlab.com/flarenetwork/coreth/params"
 )

 // memoryGasCost calculates the quadratic gas for memory expansion. It does so
diff --git a/core/vm/gas_table_test.go b/core/vm/gas_table_test.go
index 92d5d30..2150bb2 100644
--- a/core/vm/gas_table_test.go
+++ b/core/vm/gas_table_test.go
@@ -31,11 +31,11 @@ import (
        "math/big"
        "testing"

-       "github.com/ava-labs/coreth/core/rawdb"
-       "github.com/ava-labs/coreth/core/state"
-       "github.com/ava-labs/coreth/params"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/common/hexutil"
+       "gitlab.com/flarenetwork/coreth/core/rawdb"
+       "gitlab.com/flarenetwork/coreth/core/state"
+       "gitlab.com/flarenetwork/coreth/params"
 )

 func TestMemoryGasCost(t *testing.T) {
diff --git a/core/vm/instructions.go b/core/vm/instructions.go
index 81f3d3e..3fe1c4b 100644
--- a/core/vm/instructions.go
+++ b/core/vm/instructions.go
@@ -29,10 +29,10 @@ package vm
 import (
        "errors"

-       "github.com/ava-labs/coreth/core/types"
-       "github.com/ava-labs/coreth/params"
        "github.com/ethereum/go-ethereum/common"
        "github.com/holiman/uint256"
+       "gitlab.com/flarenetwork/coreth/core/types"
+       "gitlab.com/flarenetwork/coreth/params"
        "golang.org/x/crypto/sha3"
 )

diff --git a/core/vm/instructions_test.go b/core/vm/instructions_test.go
index 1570b36..50fb6db 100644
--- a/core/vm/instructions_test.go
+++ b/core/vm/instructions_test.go
@@ -33,10 +33,10 @@ import (
        "io/ioutil"
        "testing"

-       "github.com/ava-labs/coreth/params"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/crypto"
        "github.com/holiman/uint256"
+       "gitlab.com/flarenetwork/coreth/params"
 )

 type TwoOperandTestcase struct {
diff --git a/core/vm/interface.go b/core/vm/interface.go
index bde4b08..6148970 100644
--- a/core/vm/interface.go
+++ b/core/vm/interface.go
@@ -29,8 +29,8 @@ package vm
 import (
        "math/big"

-       "github.com/ava-labs/coreth/core/types"
        "github.com/ethereum/go-ethereum/common"
+       "gitlab.com/flarenetwork/coreth/core/types"
 )

 // StateDB is an EVM database for full state querying.
diff --git a/core/vm/jump_table.go b/core/vm/jump_table.go
index d833e52..249736d 100644
--- a/core/vm/jump_table.go
+++ b/core/vm/jump_table.go
@@ -27,7 +27,7 @@
 package vm

 import (
-       "github.com/ava-labs/coreth/params"
+       "gitlab.com/flarenetwork/coreth/params"
 )

 type (
diff --git a/core/vm/logger.go b/core/vm/logger.go
index f8d088d..e49d4f7 100644
--- a/core/vm/logger.go
+++ b/core/vm/logger.go
@@ -34,12 +34,12 @@ import (
        "strings"
```

```diff
        "time"

-       "github.com/ava-labs/coreth/core/types"
-       "github.com/ava-labs/coreth/params"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/common/hexutil"
        "github.com/ethereum/go-ethereum/common/math"
        "github.com/holiman/uint256"
+       "gitlab.com/flarenetwork/coreth/core/types"
+       "gitlab.com/flarenetwork/coreth/params"
 )

 // Storage represents a contract's storage.
diff --git a/core/vm/logger_test.go b/core/vm/logger_test.go
index 1d87e46..a35c4e6 100644
--- a/core/vm/logger_test.go
+++ b/core/vm/logger_test.go
@@ -30,10 +30,10 @@ import (
        "math/big"
        "testing"

-       "github.com/ava-labs/coreth/core/state"
-       "github.com/ava-labs/coreth/params"
        "github.com/ethereum/go-ethereum/common"
        "github.com/holiman/uint256"
+       "gitlab.com/flarenetwork/coreth/core/state"
+       "gitlab.com/flarenetwork/coreth/params"
 )

 type dummyContractRef struct {
diff --git a/core/vm/operations_acl.go b/core/vm/operations_acl.go
index adfe772..d9e2f06 100644
--- a/core/vm/operations_acl.go
+++ b/core/vm/operations_acl.go
@@ -29,9 +29,9 @@ package vm
 import (
        "errors"

-       "github.com/ava-labs/coreth/params"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/common/math"
+       "gitlab.com/flarenetwork/coreth/params"
 )

 // gasSStoreEIP2929 implements gas cost for SSTORE according to EIP-2929
diff --git a/core/vm/runtime/env.go b/core/vm/runtime/env.go
index 5cae37d..6ebd9c6 100644
--- a/core/vm/runtime/env.go
+++ b/core/vm/runtime/env.go
@@ -27,8 +27,8 @@
 package runtime

 import (
-       "github.com/ava-labs/coreth/core"
-       "github.com/ava-labs/coreth/core/vm"
+       "gitlab.com/flarenetwork/coreth/core"
+       "gitlab.com/flarenetwork/coreth/core/vm"
 )

 func NewEnv(cfg *Config) *vm.EVM {
diff --git a/core/vm/runtime/runtime.go b/core/vm/runtime/runtime.go
index eb6e3f6..bd24dba 100644
--- a/core/vm/runtime/runtime.go
+++ b/core/vm/runtime/runtime.go
@@ -31,12 +31,12 @@ import (
        "math/big"
        "time"

-       "github.com/ava-labs/coreth/core/rawdb"
-       "github.com/ava-labs/coreth/core/state"
-       "github.com/ava-labs/coreth/core/vm"
-       "github.com/ava-labs/coreth/params"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/crypto"
+       "gitlab.com/flarenetwork/coreth/core/rawdb"
+       "gitlab.com/flarenetwork/coreth/core/state"
+       "gitlab.com/flarenetwork/coreth/core/vm"
+       "gitlab.com/flarenetwork/coreth/params"
 )

 // Config is a basic type specifying certain configuration flags for running
diff --git a/core/vm/runtime/runtime_example_test.go b/core/vm/runtime/runtime_example_test.go
index 9850e28..c126459 100644
--- a/core/vm/runtime/runtime_example_test.go
+++ b/core/vm/runtime/runtime_example_test.go
@@ -29,8 +29,8 @@ package runtime_test
 import (
        "fmt"

-       "github.com/ava-labs/coreth/core/vm/runtime"
        "github.com/ethereum/go-ethereum/common"
+       "gitlab.com/flarenetwork/coreth/core/vm/runtime"
 )

 func ExampleExecute() {
diff --git a/core/vm/runtime/runtime_test.go b/core/vm/runtime/runtime_test.go
index 4b8d0a9..8561e31 100644
--- a/core/vm/runtime/runtime_test.go
+++ b/core/vm/runtime/runtime_test.go
@@ -34,16 +34,16 @@ import (
        "testing"
        "time"

-       "github.com/ava-labs/coreth/consensus"
-       "github.com/ava-labs/coreth/core"
-       "github.com/ava-labs/coreth/core/rawdb"
-       "github.com/ava-labs/coreth/core/state"
-       "github.com/ava-labs/coreth/core/types"
-       "github.com/ava-labs/coreth/core/vm"
-       "github.com/ava-labs/coreth/params"
        "github.com/ethereum/go-ethereum/accounts/abi"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/core/asm"
+       "gitlab.com/flarenetwork/coreth/consensus"
+       "gitlab.com/flarenetwork/coreth/core"
+       "gitlab.com/flarenetwork/coreth/core/rawdb"
+       "gitlab.com/flarenetwork/coreth/core/state"
+       "gitlab.com/flarenetwork/coreth/core/types"
+       "gitlab.com/flarenetwork/coreth/core/vm"
+       "gitlab.com/flarenetwork/coreth/params"
 )

 func TestDefaults(t *testing.T) {
diff --git a/core/vm/stack_table.go b/core/vm/stack_table.go
index 487acae..e75f21a 100644
--- a/core/vm/stack_table.go
+++ b/core/vm/stack_table.go
@@ -27,7 +27,7 @@
 package vm
 import (
-       "github.com/ava-labs/coreth/params"
+       "gitlab.com/flarenetwork/coreth/params"
 )

 func minSwapStack(n int) int {
diff --git a/eth/api.go b/eth/api.go
index 62dd67b..f0d2bce 100644
--- a/eth/api.go
+++ b/eth/api.go
@@ -35,16 +35,16 @@ import (
        "os"
        "strings"

-       "github.com/ava-labs/coreth/core"
-       "github.com/ava-labs/coreth/core/rawdb"
-       "github.com/ava-labs/coreth/core/state"
-       "github.com/ava-labs/coreth/core/types"
-       "github.com/ava-labs/coreth/internal/ethapi"
-       "github.com/ava-labs/coreth/rpc"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/common/hexutil"
        "github.com/ethereum/go-ethereum/rlp"
        "github.com/ethereum/go-ethereum/trie"
+       "gitlab.com/flarenetwork/coreth/core"
+       "gitlab.com/flarenetwork/coreth/core/rawdb"
+       "gitlab.com/flarenetwork/coreth/core/state"
+       "gitlab.com/flarenetwork/coreth/core/types"
+       "gitlab.com/flarenetwork/coreth/internal/ethapi"
+       "gitlab.com/flarenetwork/coreth/rpc"
 )

 // PublicEthereumAPI provides an API to access Ethereum full node-related
diff --git a/eth/api_backend.go b/eth/api_backend.go
index 5152b20..29e81d9 100644
--- a/eth/api_backend.go
+++ b/eth/api_backend.go
@@ -32,21 +32,21 @@ import (
        "math/big"
        "time"

-       "github.com/ava-labs/coreth/accounts"
-       "github.com/ava-labs/coreth/consensus"
-       "github.com/ava-labs/coreth/core"
-       "github.com/ava-labs/coreth/core/bloombits"
-       "github.com/ava-labs/coreth/core/rawdb"
-       "github.com/ava-labs/coreth/core/state"
-       "github.com/ava-labs/coreth/core/types"
-       "github.com/ava-labs/coreth/core/vm"
-       "github.com/ava-labs/coreth/eth/gasprice"
-       "github.com/ava-labs/coreth/params"
-       "github.com/ava-labs/coreth/rpc"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/eth/downloader"
        "github.com/ethereum/go-ethereum/ethdb"
        "github.com/ethereum/go-ethereum/event"
+       "gitlab.com/flarenetwork/coreth/accounts"
+       "gitlab.com/flarenetwork/coreth/consensus"
+       "gitlab.com/flarenetwork/coreth/core"
+       "gitlab.com/flarenetwork/coreth/core/bloombits"
+       "gitlab.com/flarenetwork/coreth/core/rawdb"
+       "gitlab.com/flarenetwork/coreth/core/state"
+       "gitlab.com/flarenetwork/coreth/core/types"
+       "gitlab.com/flarenetwork/coreth/core/vm"
+       "gitlab.com/flarenetwork/coreth/eth/gasprice"
+       "gitlab.com/flarenetwork/coreth/params"
+       "gitlab.com/flarenetwork/coreth/rpc"
 )

 var (
diff --git a/eth/backend.go b/eth/backend.go
index 380548e..8c32ac2 100644
--- a/eth/backend.go
+++ b/eth/backend.go
@@ -33,28 +33,28 @@ import (
        "sync"
        "time"

-       "github.com/ava-labs/coreth/accounts"
-       "github.com/ava-labs/coreth/consensus"
-       "github.com/ava-labs/coreth/consensus/dummy"
-       "github.com/ava-labs/coreth/core"
-       "github.com/ava-labs/coreth/core/bloombits"
-       "github.com/ava-labs/coreth/core/rawdb"
-       "github.com/ava-labs/coreth/core/types"
-       "github.com/ava-labs/coreth/core/vm"
-       "github.com/ava-labs/coreth/eth/ethconfig"
-       "github.com/ava-labs/coreth/eth/filters"
-       "github.com/ava-labs/coreth/eth/gasprice"
-       "github.com/ava-labs/coreth/eth/tracers"
-       "github.com/ava-labs/coreth/internal/ethapi"
-       "github.com/ava-labs/coreth/miner"
-       "github.com/ava-labs/coreth/node"
-       "github.com/ava-labs/coreth/params"
-       "github.com/ava-labs/coreth/rpc"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/eth/downloader"
        "github.com/ethereum/go-ethereum/ethdb"
        "github.com/ethereum/go-ethereum/event"
        "github.com/ethereum/go-ethereum/log"
+       "gitlab.com/flarenetwork/coreth/accounts"
+       "gitlab.com/flarenetwork/coreth/consensus"
+       "gitlab.com/flarenetwork/coreth/consensus/dummy"
+       "gitlab.com/flarenetwork/coreth/core"
+       "gitlab.com/flarenetwork/coreth/core/bloombits"
+       "gitlab.com/flarenetwork/coreth/core/rawdb"
+       "gitlab.com/flarenetwork/coreth/core/types"
+       "gitlab.com/flarenetwork/coreth/core/vm"
+       "gitlab.com/flarenetwork/coreth/eth/ethconfig"
+       "gitlab.com/flarenetwork/coreth/eth/filters"
+       "gitlab.com/flarenetwork/coreth/eth/gasprice"
+       "gitlab.com/flarenetwork/coreth/eth/tracers"
+       "gitlab.com/flarenetwork/coreth/internal/ethapi"
+       "gitlab.com/flarenetwork/coreth/miner"
+       "gitlab.com/flarenetwork/coreth/node"
+       "gitlab.com/flarenetwork/coreth/params"
+       "gitlab.com/flarenetwork/coreth/rpc"
 )

 // Config contains the configuration options of the ETH protocol.
diff --git a/eth/bloombits.go b/eth/bloombits.go
index ecc0aaf..b45ed5b 100644
--- a/eth/bloombits.go
+++ b/eth/bloombits.go
@@ -29,8 +29,8 @@ package eth
 import (
        "time"

-       "github.com/ava-labs/coreth/core/rawdb"
        "github.com/ethereum/go-ethereum/common/bitutil"
```

```diff
+	"gitlab.com/flarenetwork/coreth/core/rawdb"
 )
 
 const (
diff --git a/eth/ethconfig/config.go b/eth/ethconfig/config.go
index 99ea015..bbfa10b 100644
--- a/eth/ethconfig/config.go
+++ b/eth/ethconfig/config.go
@@ -29,10 +29,10 @@ package ethconfig
 import (
 	"time"
 
-	"github.com/ava-labs/coreth/core"
-	"github.com/ava-labs/coreth/eth/gasprice"
-	"github.com/ava-labs/coreth/miner"
 	"github.com/ethereum/go-ethereum/common"
+	"gitlab.com/flarenetwork/coreth/core"
+	"gitlab.com/flarenetwork/coreth/eth/gasprice"
+	"gitlab.com/flarenetwork/coreth/miner"
 )
 
 // DefaultFullGPOConfig contains default gasprice oracle settings for full node.
diff --git a/eth/filters/api.go b/eth/filters/api.go
index f543d6e..50339df 100644
--- a/eth/filters/api.go
+++ b/eth/filters/api.go
@@ -35,13 +35,13 @@ import (
 	"sync"
 	"time"
 
-	"github.com/ava-labs/coreth/core/types"
-	"github.com/ava-labs/coreth/interfaces"
-	"github.com/ava-labs/coreth/rpc"
 	"github.com/ethereum/go-ethereum/common"
 	"github.com/ethereum/go-ethereum/common/hexutil"
 	"github.com/ethereum/go-ethereum/ethdb"
 	"github.com/ethereum/go-ethereum/event"
+	"gitlab.com/flarenetwork/coreth/core/types"
+	"gitlab.com/flarenetwork/coreth/interfaces"
+	"gitlab.com/flarenetwork/coreth/rpc"
 )
 
 // filter is a helper struct that holds meta information over the filter type
diff --git a/eth/filters/api_test.go b/eth/filters/api_test.go
index 6f35639..2903c0b 100644
--- a/eth/filters/api_test.go
+++ b/eth/filters/api_test.go
@@ -21,8 +21,8 @@ import (
 	"fmt"
 	"testing"
 
-	"github.com/ava-labs/coreth/rpc"
 	"github.com/ethereum/go-ethereum/common"
+	"gitlab.com/flarenetwork/coreth/rpc"
 )
 
 func TestUnmarshalJSONNewFilterArgs(t *testing.T) {
diff --git a/eth/filters/filter.go b/eth/filters/filter.go
index 94769a1..e8f9ee7 100644
--- a/eth/filters/filter.go
+++ b/eth/filters/filter.go
@@ -32,15 +32,15 @@ import (
 	"fmt"
 	"math/big"
 
-	"github.com/ava-labs/coreth/core/vm"
+	"gitlab.com/flarenetwork/coreth/core/vm"
 
-	"github.com/ava-labs/coreth/core"
-	"github.com/ava-labs/coreth/core/bloombits"
-	"github.com/ava-labs/coreth/core/types"
-	"github.com/ava-labs/coreth/rpc"
 	"github.com/ethereum/go-ethereum/common"
 	"github.com/ethereum/go-ethereum/ethdb"
 	"github.com/ethereum/go-ethereum/event"
+	"gitlab.com/flarenetwork/coreth/core"
+	"gitlab.com/flarenetwork/coreth/core/bloombits"
+	"gitlab.com/flarenetwork/coreth/core/types"
+	"gitlab.com/flarenetwork/coreth/rpc"
 )
 
 type Backend interface {
diff --git a/eth/filters/filter_system.go b/eth/filters/filter_system.go
index 86f5a0b..db72584 100644
--- a/eth/filters/filter_system.go
+++ b/eth/filters/filter_system.go
@@ -34,14 +34,14 @@ import (
 	"sync"
 	"time"
 
-	"github.com/ava-labs/coreth/core"
-	"github.com/ava-labs/coreth/core/rawdb"
-	"github.com/ava-labs/coreth/core/types"
-	"github.com/ava-labs/coreth/interfaces"
-	"github.com/ava-labs/coreth/rpc"
 	"github.com/ethereum/go-ethereum/common"
 	"github.com/ethereum/go-ethereum/event"
 	"github.com/ethereum/go-ethereum/log"
+	"gitlab.com/flarenetwork/coreth/core"
+	"gitlab.com/flarenetwork/coreth/core/rawdb"
+	"gitlab.com/flarenetwork/coreth/core/types"
+	"gitlab.com/flarenetwork/coreth/interfaces"
+	"gitlab.com/flarenetwork/coreth/rpc"
 )
 
 // Type determines the kind of filter and is used to put the filter in to
diff --git a/eth/gasprice/gasprice.go b/eth/gasprice/gasprice.go
index b1d415b..954b124 100644
--- a/eth/gasprice/gasprice.go
+++ b/eth/gasprice/gasprice.go
@@ -33,12 +33,12 @@ import (
 	"sync"
 
 	"github.com/ava-labs/avalanchego/utils/timer"
-	"github.com/ava-labs/coreth/consensus/dummy"
-	"github.com/ava-labs/coreth/core/types"
-	"github.com/ava-labs/coreth/params"
-	"github.com/ava-labs/coreth/rpc"
 	"github.com/ethereum/go-ethereum/common"
 	"github.com/ethereum/go-ethereum/log"
+	"gitlab.com/flarenetwork/coreth/consensus/dummy"
+	"gitlab.com/flarenetwork/coreth/core/types"
+	"gitlab.com/flarenetwork/coreth/params"
+	"gitlab.com/flarenetwork/coreth/rpc"
 )
 
 const sampleNumber = 3 // Number of transactions sampled in a block
diff --git a/eth/gasprice/gasprice_test.go b/eth/gasprice/gasprice_test.go
index cfea2d4..97733eb 100644
--- a/eth/gasprice/gasprice_test.go
```

```
+++ b/eth/gasprice/gasprice_test.go
@@ -32,15 +32,15 @@ import (
        "math/big"
        "testing"

-       "github.com/ava-labs/coreth/consensus/dummy"
-       "github.com/ava-labs/coreth/core"
-       "github.com/ava-labs/coreth/core/rawdb"
-       "github.com/ava-labs/coreth/core/types"
-       "github.com/ava-labs/coreth/core/vm"
-       "github.com/ava-labs/coreth/params"
-       "github.com/ava-labs/coreth/rpc"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/crypto"
+       "gitlab.com/flarenetwork/coreth/consensus/dummy"
+       "gitlab.com/flarenetwork/coreth/core"
+       "gitlab.com/flarenetwork/coreth/core/rawdb"
+       "gitlab.com/flarenetwork/coreth/core/types"
+       "gitlab.com/flarenetwork/coreth/core/vm"
+       "gitlab.com/flarenetwork/coreth/params"
+       "gitlab.com/flarenetwork/coreth/rpc"
 )

 type testBackend struct {
diff --git a/eth/state_accessor.go b/eth/state_accessor.go
index e94b2d4..adad3d1 100644
--- a/eth/state_accessor.go
+++ b/eth/state_accessor.go
@@ -32,13 +32,13 @@ import (
        "math/big"
        "time"

-       "github.com/ava-labs/coreth/core"
-       "github.com/ava-labs/coreth/core/state"
-       "github.com/ava-labs/coreth/core/types"
-       "github.com/ava-labs/coreth/core/vm"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/log"
        "github.com/ethereum/go-ethereum/trie"
+       "gitlab.com/flarenetwork/coreth/core"
+       "gitlab.com/flarenetwork/coreth/core/state"
+       "gitlab.com/flarenetwork/coreth/core/types"
+       "gitlab.com/flarenetwork/coreth/core/vm"
 )

 // stateAtBlock retrieves the state database associated with a certain block.
diff --git a/eth/tracers/api.go b/eth/tracers/api.go
index c54cd93..46e7505 100644
--- a/eth/tracers/api.go
+++ b/eth/tracers/api.go
@@ -36,19 +36,19 @@ import (
        "sync"
        "time"

-       "github.com/ava-labs/coreth/consensus"
-       "github.com/ava-labs/coreth/core"
-       "github.com/ava-labs/coreth/core/state"
-       "github.com/ava-labs/coreth/core/types"
-       "github.com/ava-labs/coreth/core/vm"
-       "github.com/ava-labs/coreth/internal/ethapi"
-       "github.com/ava-labs/coreth/params"
-       "github.com/ava-labs/coreth/rpc"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/common/hexutil"
        "github.com/ethereum/go-ethereum/ethdb"
        "github.com/ethereum/go-ethereum/log"
        "github.com/ethereum/go-ethereum/rlp"
+       "gitlab.com/flarenetwork/coreth/consensus"
+       "gitlab.com/flarenetwork/coreth/core"
+       "gitlab.com/flarenetwork/coreth/core/state"
+       "gitlab.com/flarenetwork/coreth/core/types"
+       "gitlab.com/flarenetwork/coreth/core/vm"
+       "gitlab.com/flarenetwork/coreth/internal/ethapi"
+       "gitlab.com/flarenetwork/coreth/params"
+       "gitlab.com/flarenetwork/coreth/rpc"
 )

 const (
diff --git a/eth/tracers/api_test.go b/eth/tracers/api_test.go
index 454f35c..4122c90 100644
--- a/eth/tracers/api_test.go
+++ b/eth/tracers/api_test.go
@@ -38,20 +38,20 @@ import (
        "sort"
        "testing"

-       "github.com/ava-labs/coreth/consensus"
-       "github.com/ava-labs/coreth/consensus/dummy"
-       "github.com/ava-labs/coreth/core"
-       "github.com/ava-labs/coreth/core/rawdb"
-       "github.com/ava-labs/coreth/core/state"
-       "github.com/ava-labs/coreth/core/types"
-       "github.com/ava-labs/coreth/core/vm"
-       "github.com/ava-labs/coreth/internal/ethapi"
-       "github.com/ava-labs/coreth/params"
-       "github.com/ava-labs/coreth/rpc"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/common/hexutil"
        "github.com/ethereum/go-ethereum/crypto"
        "github.com/ethereum/go-ethereum/ethdb"
+       "gitlab.com/flarenetwork/coreth/consensus"
+       "gitlab.com/flarenetwork/coreth/consensus/dummy"
+       "gitlab.com/flarenetwork/coreth/core"
+       "gitlab.com/flarenetwork/coreth/core/rawdb"
+       "gitlab.com/flarenetwork/coreth/core/state"
+       "gitlab.com/flarenetwork/coreth/core/types"
+       "gitlab.com/flarenetwork/coreth/core/vm"
+       "gitlab.com/flarenetwork/coreth/internal/ethapi"
+       "gitlab.com/flarenetwork/coreth/params"
+       "gitlab.com/flarenetwork/coreth/rpc"
 )

 var (
diff --git a/eth/tracers/tracer.go b/eth/tracers/tracer.go
index 3f4325a..4670281 100644
--- a/eth/tracers/tracer.go
+++ b/eth/tracers/tracer.go
@@ -35,12 +35,12 @@ import (
        "time"
        "unsafe"

-       "github.com/ava-labs/coreth/core"
-       "github.com/ava-labs/coreth/core/vm"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/common/hexutil"
        "github.com/ethereum/go-ethereum/crypto"
        "github.com/ethereum/go-ethereum/log"
+       "gitlab.com/flarenetwork/coreth/core"
+       "gitlab.com/flarenetwork/coreth/core/vm"
        "gopkg.in/olebedev/go-duktape.v3"
 )

diff --git a/eth/tracers/tracer_test.go b/eth/tracers/tracer_test.go
index 3d4e94c..e8ce389 100644
--- a/eth/tracers/tracer_test.go
+++ b/eth/tracers/tracer_test.go
@@ -33,10 +33,10 @@ import (
        "testing"
        "time"

-       "github.com/ava-labs/coreth/core/state"
-       "github.com/ava-labs/coreth/core/vm"
-       "github.com/ava-labs/coreth/params"
        "github.com/ethereum/go-ethereum/common"
+       "gitlab.com/flarenetwork/coreth/core/state"
+       "gitlab.com/flarenetwork/coreth/core/vm"
+       "gitlab.com/flarenetwork/coreth/params"
 )

 type account struct{}
diff --git a/eth/tracers/tracers.go b/eth/tracers/tracers.go
index dbe53f3..049cb18 100644
--- a/eth/tracers/tracers.go
+++ b/eth/tracers/tracers.go
@@ -31,7 +31,7 @@ import (
        "strings"
        "unicode"

-       "github.com/ava-labs/coreth/eth/tracers/internal/tracers"
+       "gitlab.com/flarenetwork/coreth/eth/tracers/internal/tracers"
 )

 // all contains all the built in JavaScript tracers by name.
diff --git a/eth/tracers/tracers_test.go b/eth/tracers/tracers_test.go
index f9e55e2..ded3c86 100644
--- a/eth/tracers/tracers_test.go
+++ b/eth/tracers/tracers_test.go
@@ -37,17 +37,17 @@ import (
        "strings"
        "testing"

-       "github.com/ava-labs/coreth/core"
-       "github.com/ava-labs/coreth/core/rawdb"
-       "github.com/ava-labs/coreth/core/types"
-       "github.com/ava-labs/coreth/core/vm"
-       "github.com/ava-labs/coreth/params"
-       "github.com/ava-labs/coreth/tests"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/common/hexutil"
        "github.com/ethereum/go-ethereum/common/math"
        "github.com/ethereum/go-ethereum/crypto"
        "github.com/ethereum/go-ethereum/rlp"
+       "gitlab.com/flarenetwork/coreth/core"
+       "gitlab.com/flarenetwork/coreth/core/rawdb"
+       "gitlab.com/flarenetwork/coreth/core/types"
+       "gitlab.com/flarenetwork/coreth/core/vm"
+       "gitlab.com/flarenetwork/coreth/params"
+       "gitlab.com/flarenetwork/coreth/tests"
 )

 // To generate a new callTracer test, copy paste the makeTest method below into
diff --git a/ethclient/corethclient/corethclient.go b/ethclient/corethclient/corethclient.go
index 8d7654f..da9ca7f 100644
--- a/ethclient/corethclient/corethclient.go
+++ b/ethclient/corethclient/corethclient.go
@@ -33,12 +33,12 @@ import (
        "runtime"
        "runtime/debug"

-       "github.com/ava-labs/coreth/core/types"
-       "github.com/ava-labs/coreth/ethclient"
-       "github.com/ava-labs/coreth/interfaces"
-       "github.com/ava-labs/coreth/rpc"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/common/hexutil"
+       "gitlab.com/flarenetwork/coreth/core/types"
+       "gitlab.com/flarenetwork/coreth/ethclient"
+       "gitlab.com/flarenetwork/coreth/interfaces"
+       "gitlab.com/flarenetwork/coreth/rpc"
 )

 // Client is a wrapper around rpc.Client that implements geth-specific functionality.
diff --git a/ethclient/ethclient.go b/ethclient/ethclient.go
index 0959a0a..8d30ca2 100644
--- a/ethclient/ethclient.go
+++ b/ethclient/ethclient.go
@@ -35,11 +35,11 @@ import (
        "math/big"

        "github.com/ava-labs/avalanchego/ids"
-       "github.com/ava-labs/coreth/core/types"
-       "github.com/ava-labs/coreth/interfaces"
-       "github.com/ava-labs/coreth/rpc"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/common/hexutil"
+       "gitlab.com/flarenetwork/coreth/core/types"
+       "gitlab.com/flarenetwork/coreth/interfaces"
+       "gitlab.com/flarenetwork/coreth/rpc"
 )

 // Client defines typed wrappers for the Ethereum RPC API.
diff --git a/ethclient/signer.go b/ethclient/signer.go
index dafa943..45ef52d 100644
--- a/ethclient/signer.go
+++ b/ethclient/signer.go
@@ -30,8 +30,8 @@ import (
        "errors"
        "math/big"

-       "github.com/ava-labs/coreth/core/types"
        "github.com/ethereum/go-ethereum/common"
+       "gitlab.com/flarenetwork/coreth/core/types"
 )

 // senderFromServer is a types.Signer that remembers the sender address returned by the RPC
diff --git a/go.mod b/go.mod
index 981a551..fac1e12 100644
--- a/go.mod
+++ b/go.mod
@@ -1,4 +1,4 @@
-module github.com/ava-labs/coreth
+module gitlab.com/flarenetwork/coreth

 go 1.15

diff --git a/interfaces/interfaces.go b/interfaces/interfaces.go
index efcde08..be8870f 100644
--- a/interfaces/interfaces.go
+++ b/interfaces/interfaces.go
@@ -32,8 +32,8 @@ import (
 	"errors"
 	"math/big"
 
-	"github.com/ava-labs/coreth/core/types"
 	"github.com/ethereum/go-ethereum/common"
+	"gitlab.com/flarenetwork/coreth/core/types"
 )
 
 // NotFound is returned by API methods if the requested item does not exist.
diff --git a/internal/ethapi/api.go b/internal/ethapi/api.go
index b9ad558..e110a19 100644
--- a/internal/ethapi/api.go
+++ b/internal/ethapi/api.go
@@ -35,15 +35,6 @@ import (
 	"time"
 
 	"github.com/ava-labs/avalanchego/ids"
-	"github.com/ava-labs/coreth/accounts"
-	"github.com/ava-labs/coreth/accounts/keystore"
-	"github.com/ava-labs/coreth/accounts/scwallet"
-	"github.com/ava-labs/coreth/core"
-	"github.com/ava-labs/coreth/core/state"
-	"github.com/ava-labs/coreth/core/types"
-	"github.com/ava-labs/coreth/core/vm"
-	"github.com/ava-labs/coreth/params"
-	"github.com/ava-labs/coreth/rpc"
 	"github.com/davecgh/go-spew/spew"
 	"github.com/ethereum/go-ethereum/accounts/abi"
 	"github.com/ethereum/go-ethereum/common"
@@ -53,6 +44,15 @@ import (
 	"github.com/ethereum/go-ethereum/log"
 	"github.com/ethereum/go-ethereum/rlp"
 	"github.com/tyler-smith/go-bip39"
+	"gitlab.com/flarenetwork/coreth/accounts"
+	"gitlab.com/flarenetwork/coreth/accounts/keystore"
+	"gitlab.com/flarenetwork/coreth/accounts/scwallet"
+	"gitlab.com/flarenetwork/coreth/core"
+	"gitlab.com/flarenetwork/coreth/core/state"
+	"gitlab.com/flarenetwork/coreth/core/types"
+	"gitlab.com/flarenetwork/coreth/core/vm"
+	"gitlab.com/flarenetwork/coreth/params"
+	"gitlab.com/flarenetwork/coreth/rpc"
 )
 
 // PublicEthereumAPI provides an API to access Ethereum related information.
diff --git a/internal/ethapi/backend.go b/internal/ethapi/backend.go
index 3d03734..f3b1643 100644
--- a/internal/ethapi/backend.go
+++ b/internal/ethapi/backend.go
@@ -31,19 +31,19 @@ import (
 	"context"
 	"math/big"
 
-	"github.com/ava-labs/coreth/accounts"
-	"github.com/ava-labs/coreth/consensus"
-	"github.com/ava-labs/coreth/core"
-	"github.com/ava-labs/coreth/core/bloombits"
-	"github.com/ava-labs/coreth/core/state"
-	"github.com/ava-labs/coreth/core/types"
-	"github.com/ava-labs/coreth/core/vm"
-	"github.com/ava-labs/coreth/params"
-	"github.com/ava-labs/coreth/rpc"
 	"github.com/ethereum/go-ethereum/common"
 	"github.com/ethereum/go-ethereum/eth/downloader"
 	"github.com/ethereum/go-ethereum/ethdb"
 	"github.com/ethereum/go-ethereum/event"
+	"gitlab.com/flarenetwork/coreth/accounts"
+	"gitlab.com/flarenetwork/coreth/consensus"
+	"gitlab.com/flarenetwork/coreth/core"
+	"gitlab.com/flarenetwork/coreth/core/bloombits"
+	"gitlab.com/flarenetwork/coreth/core/state"
+	"gitlab.com/flarenetwork/coreth/core/types"
+	"gitlab.com/flarenetwork/coreth/core/vm"
+	"gitlab.com/flarenetwork/coreth/params"
+	"gitlab.com/flarenetwork/coreth/rpc"
 )
 
 // Backend interface provides the common API services (that are provided by
diff --git a/internal/ethapi/transaction_args.go b/internal/ethapi/transaction_args.go
index 6b06742..dcadfd7 100644
--- a/internal/ethapi/transaction_args.go
+++ b/internal/ethapi/transaction_args.go
@@ -33,12 +33,12 @@ import (
 	"fmt"
 	"math/big"
 
-	"github.com/ava-labs/coreth/core/types"
-	"github.com/ava-labs/coreth/rpc"
 	"github.com/ethereum/go-ethereum/common"
 	"github.com/ethereum/go-ethereum/common/hexutil"
 	"github.com/ethereum/go-ethereum/common/math"
 	"github.com/ethereum/go-ethereum/log"
+	"gitlab.com/flarenetwork/coreth/core/types"
+	"gitlab.com/flarenetwork/coreth/rpc"
 )
 
 // TransactionArgs represents the arguments to construct a new transaction
diff --git a/miner/miner.go b/miner/miner.go
index 263439c..24ddc22 100644
--- a/miner/miner.go
+++ b/miner/miner.go
@@ -28,12 +28,12 @@
 package miner
 
 import (
-	"github.com/ava-labs/coreth/consensus"
-	"github.com/ava-labs/coreth/core"
-	"github.com/ava-labs/coreth/core/types"
-	"github.com/ava-labs/coreth/params"
 	"github.com/ethereum/go-ethereum/common"
 	"github.com/ethereum/go-ethereum/event"
+	"gitlab.com/flarenetwork/coreth/consensus"
+	"gitlab.com/flarenetwork/coreth/core"
+	"gitlab.com/flarenetwork/coreth/core/types"
+	"gitlab.com/flarenetwork/coreth/params"
 )
 
 // Backend wraps all methods required for mining.
diff --git a/miner/worker.go b/miner/worker.go
index c8c7268..69fa1f6 100644
--- a/miner/worker.go
+++ b/miner/worker.go
@@ -36,16 +36,16 @@ import (
 	"sync"
 	"time"
 
-	"github.com/ava-labs/coreth/consensus"
-	"github.com/ava-labs/coreth/consensus/dummy"
-	"github.com/ava-labs/coreth/consensus/misc"
-	"github.com/ava-labs/coreth/core"
```

```
-       "github.com/ava-labs/coreth/core/state"
-       "github.com/ava-labs/coreth/core/types"
-       "github.com/ava-labs/coreth/params"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/event"
        "github.com/ethereum/go-ethereum/log"
+       "gitlab.com/flarenetwork/coreth/consensus"
+       "gitlab.com/flarenetwork/coreth/consensus/dummy"
+       "gitlab.com/flarenetwork/coreth/consensus/misc"
+       "gitlab.com/flarenetwork/coreth/core"
+       "gitlab.com/flarenetwork/coreth/core/state"
+       "gitlab.com/flarenetwork/coreth/core/types"
+       "gitlab.com/flarenetwork/coreth/params"
 )

 // environment is the worker's current environment and holds all of the current state information.
diff --git a/node/api.go b/node/api.go
index c38b2df..0c36fc7 100644
--- a/node/api.go
+++ b/node/api.go
@@ -27,10 +27,10 @@
 package node

 import (
-       "github.com/ava-labs/coreth/internal/debug"
-       "github.com/ava-labs/coreth/rpc"
        "github.com/ethereum/go-ethereum/common/hexutil"
        "github.com/ethereum/go-ethereum/crypto"
+       "gitlab.com/flarenetwork/coreth/internal/debug"
+       "gitlab.com/flarenetwork/coreth/rpc"
 )

 // apis returns the collection of built-in RPC APIs.
diff --git a/node/config.go b/node/config.go
index 4d2b83a..b51e67a 100644
--- a/node/config.go
+++ b/node/config.go
@@ -32,12 +32,12 @@ import (
        "os"
        "path/filepath"

-       "github.com/ava-labs/coreth/accounts"
-       "github.com/ava-labs/coreth/accounts/external"
-       "github.com/ava-labs/coreth/accounts/keystore"
+       "gitlab.com/flarenetwork/coreth/accounts"
+       "gitlab.com/flarenetwork/coreth/accounts/external"
+       "gitlab.com/flarenetwork/coreth/accounts/keystore"

-       "github.com/ava-labs/coreth/rpc"
        "github.com/ethereum/go-ethereum/log"
+       "gitlab.com/flarenetwork/coreth/rpc"
 )

 // Config represents a small collection of configuration values to fine tune the
diff --git a/node/defaults.go b/node/defaults.go
index e4c826b..e3e2257 100644
--- a/node/defaults.go
+++ b/node/defaults.go
@@ -27,7 +27,7 @@
 package node

 import (
-       "github.com/ava-labs/coreth/rpc"
+       "gitlab.com/flarenetwork/coreth/rpc"
 )

 const (
diff --git a/node/node.go b/node/node.go
index 265e00d..1272a9f 100644
--- a/node/node.go
+++ b/node/node.go
@@ -31,8 +31,8 @@ import (

        "github.com/ethereum/go-ethereum/event"

-       "github.com/ava-labs/coreth/accounts"
-       "github.com/ava-labs/coreth/rpc"
+       "gitlab.com/flarenetwork/coreth/accounts"
+       "gitlab.com/flarenetwork/coreth/rpc"
 )

 // Node is a container on which services can be registered.
diff --git a/plugin/evm/block.go b/plugin/evm/block.go
index cae83af..d749229 100644
--- a/plugin/evm/block.go
+++ b/plugin/evm/block.go
@@ -12,8 +12,8 @@ import (
        "github.com/ethereum/go-ethereum/log"
        "github.com/ethereum/go-ethereum/rlp"

-       "github.com/ava-labs/coreth/core/types"
-       "github.com/ava-labs/coreth/params"
+       "gitlab.com/flarenetwork/coreth/core/types"
+       "gitlab.com/flarenetwork/coreth/params"

        "github.com/ava-labs/avalanchego/ids"
        "github.com/ava-labs/avalanchego/snow/choices"
diff --git a/plugin/evm/block_verification.go b/plugin/evm/block_verification.go
index 03910af..1e52707 100644
--- a/plugin/evm/block_verification.go
+++ b/plugin/evm/block_verification.go
@@ -7,11 +7,11 @@ import (
        "fmt"
        "math/big"

-       coreth "github.com/ava-labs/coreth/chain"
-       "github.com/ava-labs/coreth/core/types"
-       "github.com/ava-labs/coreth/params"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/trie"
+       coreth "gitlab.com/flarenetwork/coreth/chain"
+       "gitlab.com/flarenetwork/coreth/core/types"
+       "gitlab.com/flarenetwork/coreth/params"
 )

 var (
diff --git a/plugin/evm/config.go b/plugin/evm/config.go
index f184a09..721bf75 100644
--- a/plugin/evm/config.go
+++ b/plugin/evm/config.go
@@ -7,8 +7,8 @@ import (
        "encoding/json"
        "time"

-       "github.com/ava-labs/coreth/eth"
        "github.com/spf13/cast"
+       "gitlab.com/flarenetwork/coreth/eth"
 )

 const (
diff --git a/plugin/evm/export_tx.go b/plugin/evm/export_tx.go
index 3b50915..be2275f 100644
--- a/plugin/evm/export_tx.go
+++ b/plugin/evm/export_tx.go
@@ -1,5 +1,12 @@
+// (c) 2021, Flare Networks Limited. All rights reserved.
+//
+// This file is a derived work, based on the avalanchego library whose original
+// notice appears below. It is distributed under a license compatible with the
+// licensing terms of the original code from which it is derived.
+// Please see the file LICENSE_AVALABS for licensing terms of the original work.
+// Please see the file LICENSE for licensing terms.
+//
 // (c) 2019-2020, Ava Labs, Inc. All rights reserved.
-// See the file LICENSE for licensing terms.
 
 package evm
 
@@ -7,19 +14,14 @@ import (
        "fmt"
        "math/big"
 
-       "github.com/ava-labs/coreth/core/state"
-       "github.com/ava-labs/coreth/params"
+       "gitlab.com/flarenetwork/coreth/core/state"
+       "gitlab.com/flarenetwork/coreth/params"
 
-       "github.com/ava-labs/avalanchego/chains/atomic"
        "github.com/ava-labs/avalanchego/database"
        "github.com/ava-labs/avalanchego/ids"
        "github.com/ava-labs/avalanchego/snow"
        "github.com/ava-labs/avalanchego/utils/crypto"
-       "github.com/ava-labs/avalanchego/utils/math"
        "github.com/ava-labs/avalanchego/vms/components/avax"
-       "github.com/ava-labs/avalanchego/vms/secp256k1fx"
-       "github.com/ethereum/go-ethereum/common"
-       "github.com/ethereum/go-ethereum/log"
 )
 
 // UnsignedExportTx is an unsigned ExportTx
@@ -46,75 +48,16 @@ func (tx *UnsignedExportTx) Verify(
        ctx *snow.Context,
        rules params.Rules,
 ) error {
-       switch {
-       case tx == nil:
-               return errNilTx
-       case tx.DestinationChain != avmID:
-               return errWrongChainID
-       case len(tx.ExportedOutputs) == 0:
-               return errNoExportOutputs
-       case tx.NetworkID != ctx.NetworkID:
-               return errWrongNetworkID
-       case ctx.ChainID != tx.BlockchainID:
-               return errWrongBlockchainID
-       }
-
-       for _, in := range tx.Ins {
-               if err := in.Verify(); err != nil {
-                       return err
-               }
-       }
-
-       for _, out := range tx.ExportedOutputs {
-               if err := out.Verify(); err != nil {
-                       return err
-               }
-       }
-       if !avax.IsSortedTransferableOutputs(tx.ExportedOutputs, Codec) {
-               return errOutputsNotSorted
-       }
-       if rules.IsApricotPhase1 && !IsSortedAndUniqueEVMInputs(tx.Ins) {
-               return errInputsNotSortedUnique
-       }
-
-       return nil
+       return errWrongChainID
 }
 
 func (tx *UnsignedExportTx) Cost() (uint64, error) {
-       byteCost := calcBytesCost(len(tx.UnsignedBytes()))
-       numSigs := uint64(len(tx.Ins))
-       sigCost, err := math.Mul64(numSigs, secp256k1fx.CostPerSignature)
-       if err != nil {
-               return 0, err
-       }
-       return math.Add64(byteCost, sigCost)
+       return 0, fmt.Errorf("exportTx transactions disabled")
 }
 
 // Amount of [assetID] burned by this transaction
 func (tx *UnsignedExportTx) Burned(assetID ids.ID) (uint64, error) {
-       var (
-               spent uint64
-               input uint64
-               err   error
-       )
-       for _, out := range tx.ExportedOutputs {
-               if out.AssetID() == assetID {
-                       spent, err = math.Add64(spent, out.Output().Amount())
-                       if err != nil {
-                               return 0, err
-                       }
-               }
-       }
-       for _, in := range tx.Ins {
-               if in.AssetID == assetID {
-                       input, err = math.Add64(input, in.Amount)
-                       if err != nil {
-                               return 0, err
-                       }
-               }
-       }
-
-       return math.Sub64(input, spent)
+       return 0, fmt.Errorf("exportTx transactions disabled")
 }
 
 // SemanticVerify this transaction is valid.
@@ -125,105 +68,12 @@ func (tx *UnsignedExportTx) SemanticVerify(
        baseFee *big.Int,
        rules params.Rules,
 ) error {
-       if err := tx.Verify(vm.ctx.XChainID, vm.ctx, rules); err != nil {
-               return err
-       }
-
-       // Check the transaction consumes and produces the right amounts
-       fc := avax.NewFlowChecker()
-       switch {
```

```
-       // Apply dynamic fees to export transactions as of Apricot Phase 3
-       case rules.IsApricotPhase3:
-               cost, err := stx.Cost()
-               if err != nil {
-                       return err
-               }
-               txFee, err := calculateDynamicFee(cost, baseFee)
-               if err != nil {
-                       return err
-               }
-               fc.Produce(vm.ctx.AVAXAssetID, txFee)
-
-       // Apply fees to export transactions before Apricot Phase 3
-       default:
-               fc.Produce(vm.ctx.AVAXAssetID, params.AvalancheAtomicTxFee)
-       }
-       for _, out := range tx.ExportedOutputs {
-               fc.Produce(out.AssetID(), out.Output().Amount())
-       }
-       for _, in := range tx.Ins {
-               fc.Consume(in.AssetID, in.Amount)
-       }
-
-       if err := fc.Verify(); err != nil {
-               return fmt.Errorf("export tx flow check failed due to: %w", err)
-       }
-
-       if len(tx.Ins) != len(stx.Creds) {
-               return fmt.Errorf("export tx contained mismatched number of inputs/credentials (%d vs. %d)", len(tx.Ins), len(stx.Creds))
-       }
-
-       for i, input := range tx.Ins {
-               cred, ok := stx.Creds[i].(*secp256k1fx.Credential)
-               if !ok {
-                       return fmt.Errorf("expected *secp256k1fx.Credential but got %T", cred)
-               }
-               if err := cred.Verify(); err != nil {
-                       return err
-               }
-
-               if len(cred.Sigs) != 1 {
-                       return fmt.Errorf("expected one signature for EVM Input Credential, but found: %d", len(cred.Sigs))
-               }
-               pubKeyIntf, err := vm.secpFactory.RecoverPublicKey(tx.UnsignedBytes(), cred.Sigs[0][:])
-               if err != nil {
-                       return err
-               }
-               pubKey, ok := pubKeyIntf.(*crypto.PublicKeySECP256K1R)
-               if !ok {
-                       // This should never happen
-                       return fmt.Errorf("expected *crypto.PublicKeySECP256K1R but got %T", pubKeyIntf)
-               }
-               if input.Address != PublicKeyToEthAddress(pubKey) {
-                       return errPublicKeySignatureMismatch
-               }
-       }
-
-       return nil
+       return fmt.Errorf("exportTx transactions disabled")
 }

 // Accept this transaction.
 func (tx *UnsignedExportTx) Accept(ctx *snow.Context, batch database.Batch) error {
-       txID := tx.ID()
-
-       elems := make([]*atomic.Element, len(tx.ExportedOutputs))
-       for i, out := range tx.ExportedOutputs {
-               utxo := &avax.UTXO{
-                       UTXOID: avax.UTXOID{
-                               TxID:        txID,
-                               OutputIndex: uint32(i),
-                       },
-                       Asset: avax.Asset{ID: out.AssetID()},
-                       Out:   out.Out,
-               }
-
-               utxoBytes, err := Codec.Marshal(codecVersion, utxo)
-               if err != nil {
-                       return err
-               }
-               utxoID := utxo.InputID()
-               elem := &atomic.Element{
-                       Key:   utxoID[:],
-                       Value: utxoBytes,
-               }
-               if out, ok := utxo.Out.(avax.Addressable); ok {
-                       elem.Traits = out.Addresses()
-               }
-
-               elems[i] = elem
-       }
-
-       return ctx.SharedMemory.Apply(map[ids.ID]*atomic.Requests{tx.DestinationChain: {PutRequests: elems}}, batch)
+       return fmt.Errorf("exportTx transactions disabled")
 }

 // newExportTx returns a new ExportTx
@@ -235,121 +85,10 @@ func (vm *VM) newExportTx(
        baseFee *big.Int, // fee to use post-AP3
        keys []*crypto.PrivateKeySECP256K1R, // Pay the fee and provide the tokens
 ) (*Tx, error) {
-       if vm.ctx.XChainID != chainID {
-               return nil, errWrongChainID
-       }
-
-       outs := []*avax.TransferableOutput{{ // Exported to X-Chain
-               Asset: avax.Asset{ID: assetID},
-               Out: &secp256k1fx.TransferOutput{
-                       Amt: amount,
-                       OutputOwners: secp256k1fx.OutputOwners{
-                               Locktime:  0,
-                               Threshold: 1,
-                               Addrs:     []ids.ShortID{to},
-                       },
-               },
-       }}
-
-       var (
-               avaxNeeded           uint64 = 0
-               ins, avaxIns         []EVMInput
-               signers, avaxSigners [][]*crypto.PrivateKeySECP256K1R
-               err                  error
-       )
-
-       // consume non-AVAX
-       if assetID != vm.ctx.AVAXAssetID {
-               ins, signers, err = vm.GetSpendableFunds(keys, assetID, amount)
-               if err != nil {
-                       return nil, fmt.Errorf("couldn't generate tx inputs/signers: %w", err)
-               }
```

```diff
-       } else {
-               avaxNeeded = amount
-       }
-
-       rules := vm.currentRules()
-       switch {
-       case rules.IsApricotPhase3:
-               utx := &UnsignedExportTx{
-                       NetworkID:        vm.ctx.NetworkID,
-                       BlockchainID:     vm.ctx.ChainID,
-                       DestinationChain: chainID,
-                       Ins:              ins,
-                       ExportedOutputs:  outs,
-               }
-               tx := &Tx{UnsignedAtomicTx: utx}
-               if err := tx.Sign(vm.codec, nil); err != nil {
-                       return nil, err
-               }
-
-               var cost uint64
-               cost, err = tx.Cost()
-               if err != nil {
-                       return nil, err
-               }
-
-               avaxIns, avaxSigners, err = vm.GetSpendableAVAXWithFee(keys, avaxNeeded, cost, baseFee)
-       default:
-               var newAvaxNeeded uint64
-               newAvaxNeeded, err = math.Add64(avaxNeeded, params.AvalancheAtomicTxFee)
-               if err != nil {
-                       return nil, errOverflowExport
-               }
-               avaxIns, avaxSigners, err = vm.GetSpendableFunds(keys, vm.ctx.AVAXAssetID, newAvaxNeeded)
-       }
-       if err != nil {
-               return nil, fmt.Errorf("couldn't generate tx inputs/signers: %w", err)
-       }
-       ins = append(ins, avaxIns...)
-       signers = append(signers, avaxSigners...)
-
-       SortEVMInputsAndSigners(ins, signers)
-
-       // Create the transaction
-       utx := &UnsignedExportTx{
-               NetworkID:        vm.ctx.NetworkID,
-               BlockchainID:     vm.ctx.ChainID,
-               DestinationChain: chainID,
-               Ins:              ins,
-               ExportedOutputs:  outs,
-       }
-       tx := &Tx{UnsignedAtomicTx: utx}
-       if err := tx.Sign(vm.codec, signers); err != nil {
-               return nil, err
-       }
-       return tx, utx.Verify(vm.ctx.XChainID, vm.ctx, vm.currentRules())
+       return nil, errWrongChainID
 }

 // EVMStateTransfer executes the state update from the atomic export transaction
 func (tx *UnsignedExportTx) EVMStateTransfer(ctx *snow.Context, state *state.StateDB) error {
-       addrs := map[[20]byte]uint64{}
-       for _, from := range tx.Ins {
-               if from.AssetID == ctx.AVAXAssetID {
-                       log.Debug("crosschain C->X", "addr", from.Address, "amount", from.Amount, "assetID", "AVAX")
-                       // We multiply the input amount by x2cRate to convert AVAX back to the appropriate
-                       // denomination before export.
-                       amount := new(big.Int).Mul(
-                               new(big.Int).SetUint64(from.Amount), x2cRate)
-                       if state.GetBalance(from.Address).Cmp(amount) < 0 {
-                               return errInsufficientFunds
-                       }
-                       state.SubBalance(from.Address, amount)
-               } else {
-                       log.Debug("crosschain C->X", "addr", from.Address, "amount", from.Amount, "assetID", from.AssetID)
-                       amount := new(big.Int).SetUint64(from.Amount)
-                       if state.GetBalanceMultiCoin(from.Address, common.Hash(from.AssetID)).Cmp(amount) < 0 {
-                               return errInsufficientFunds
-                       }
-                       state.SubBalanceMultiCoin(from.Address, common.Hash(from.AssetID), amount)
-               }
-               if state.GetNonce(from.Address) != from.Nonce {
-                       return errInvalidNonce
-               }
-               addrs[from.Address] = from.Nonce
-       }
-       for addr, nonce := range addrs {
-               state.SetNonce(addr, nonce+1)
-       }
-       return nil
+       return errInsufficientFunds
 }
diff --git a/plugin/evm/export_tx_test.go b/plugin/evm/export_tx_test.go
index 44d6b6a..03af06a 100644
--- a/plugin/evm/export_tx_test.go
+++ b/plugin/evm/export_tx_test.go
@@ -7,7 +7,7 @@ import (
        "math/big"
        "testing"

-       "github.com/ava-labs/coreth/params"
+       "gitlab.com/flarenetwork/coreth/params"

        "github.com/ava-labs/avalanchego/chains/atomic"
        "github.com/ava-labs/avalanchego/ids"
diff --git a/plugin/evm/gasprice_update.go b/plugin/evm/gasprice_update.go
index b96fa69..2c98fb2 100644
--- a/plugin/evm/gasprice_update.go
+++ b/plugin/evm/gasprice_update.go
@@ -8,7 +8,7 @@ import (
        "sync"
        "time"

-       "github.com/ava-labs/coreth/params"
+       "gitlab.com/flarenetwork/coreth/params"
 )

 type gasPriceUpdater struct {
diff --git a/plugin/evm/gasprice_update_test.go b/plugin/evm/gasprice_update_test.go
index 179dbb8..4ab77ad 100644
--- a/plugin/evm/gasprice_update_test.go
+++ b/plugin/evm/gasprice_update_test.go
@@ -9,7 +9,7 @@ import (
        "testing"
        "time"

-       "github.com/ava-labs/coreth/params"
+       "gitlab.com/flarenetwork/coreth/params"
 )

 type mockGasPriceSetter struct {
```

```
diff --git a/plugin/evm/import_tx.go b/plugin/evm/import_tx.go
index 0349820..8013ece 100644
--- a/plugin/evm/import_tx.go
+++ b/plugin/evm/import_tx.go
@@ -1,5 +1,12 @@
+// (c) 2021, Flare Networks Limited. All rights reserved.
+//
+// This file is a derived work, based on the avalanchego library whose original
+// notice appears below. It is distributed under a license compatible with the
+// licensing terms of the original code from which it is derived.
+// Please see the file LICENSE_AVALABS for licensing terms of the original work.
+// Please see the file LICENSE for licensing terms.
+//
 // (c) 2019-2020, Ava Labs, Inc. All rights reserved.
-// See the file LICENSE for licensing terms.
 
 package evm
 
@@ -7,19 +14,15 @@ import (
        "fmt"
        "math/big"
 
-       "github.com/ava-labs/coreth/core/state"
-       "github.com/ava-labs/coreth/params"
+       "gitlab.com/flarenetwork/coreth/core/state"
+       "gitlab.com/flarenetwork/coreth/params"
 
-       "github.com/ava-labs/avalanchego/chains/atomic"
        "github.com/ava-labs/avalanchego/database"
        "github.com/ava-labs/avalanchego/ids"
        "github.com/ava-labs/avalanchego/snow"
        "github.com/ava-labs/avalanchego/utils/crypto"
-       "github.com/ava-labs/avalanchego/utils/math"
        "github.com/ava-labs/avalanchego/vms/components/avax"
-       "github.com/ava-labs/avalanchego/vms/secp256k1fx"
        "github.com/ethereum/go-ethereum/common"
-       "github.com/ethereum/go-ethereum/log"
 )
 
 // UnsignedImportTx is an unsigned ImportTx
@@ -52,89 +55,16 @@ func (tx *UnsignedImportTx) Verify(
        ctx *snow.Context,
        rules params.Rules,
 ) error {
-       switch {
-       case tx == nil:
-               return errNilTx
-       case tx.SourceChain != avmID:
-               return errWrongChainID
-       case len(tx.ImportedInputs) == 0:
-               return errNoImportInputs
-       case tx.NetworkID != ctx.NetworkID:
-               return errWrongNetworkID
-       case ctx.ChainID != tx.BlockchainID:
-               return errWrongBlockchainID
-       case rules.IsApricotPhase3 && len(tx.Outs) == 0:
-               return errNoEVMOutputs
-       }
-
-       for _, out := range tx.Outs {
-               if err := out.Verify(); err != nil {
-                       return err
-               }
-       }
-
-       for _, in := range tx.ImportedInputs {
-               if err := in.Verify(); err != nil {
-                       return err
-               }
-       }
-       if !avax.IsSortedAndUniqueTransferableInputs(tx.ImportedInputs) {
-               return errInputsNotSortedUnique
-       }
-
-       if rules.IsApricotPhase2 {
-               if !IsSortedAndUniqueEVMOutputs(tx.Outs) {
-                       return errOutputsNotSortedUnique
-               }
-       } else if rules.IsApricotPhase1 {
-               if !IsSortedEVMOutputs(tx.Outs) {
-                       return errOutputsNotSorted
-               }
-       }
-
-       return nil
+       return errWrongChainID
 }
 
 func (tx *UnsignedImportTx) Cost() (uint64, error) {
-       cost := calcBytesCost(len(tx.UnsignedBytes()))
-       for _, in := range tx.ImportedInputs {
-               inCost, err := in.In.Cost()
-               if err != nil {
-                       return 0, err
-               }
-               cost, err = math.Add64(cost, inCost)
-               if err != nil {
-                       return 0, err
-               }
-       }
-       return cost, nil
+       return 0, fmt.Errorf("exportTx transactions disabled")
 }
 
 // Amount of [assetID] burned by this transaction
 func (tx *UnsignedImportTx) Burned(assetID ids.ID) (uint64, error) {
-       var (
-               spent uint64
-               input uint64
-               err   error
-       )
-       for _, out := range tx.Outs {
-               if out.AssetID == assetID {
-                       spent, err = math.Add64(spent, out.Amount)
-                       if err != nil {
-                               return 0, err
-                       }
-               }
-       }
-       for _, in := range tx.ImportedInputs {
-               if in.AssetID() == assetID {
-                       input, err = math.Add64(input, in.Input().Amount())
-                       if err != nil {
-                               return 0, err
-                       }
-               }
-       }
-
-       return math.Sub64(input, spent)
+       return 0, fmt.Errorf("exportTx transactions disabled")
```

```
 }

 // SemanticVerify this transaction is valid.
@@ -145,82 +75,7 @@ func (tx *UnsignedImportTx) SemanticVerify(
        baseFee *big.Int,
        rules params.Rules,
 ) error {
-       if err := tx.Verify(vm.ctx.XChainID, vm.ctx, rules); err != nil {
-               return err
-       }
-
-       // Check the transaction consumes and produces the right amounts
-       fc := avax.NewFlowChecker()
-       switch {
-       // Apply dynamic fees to import transactions as of Apricot Phase 3
-       case rules.IsApricotPhase3:
-               cost, err := stx.Cost()
-               if err != nil {
-                       return err
-               }
-               txFee, err := calculateDynamicFee(cost, baseFee)
-               if err != nil {
-                       return err
-               }
-               fc.Produce(vm.ctx.AVAXAssetID, txFee)
-
-       // Apply fees to import transactions as of Apricot Phase 2
-       case rules.IsApricotPhase2:
-               fc.Produce(vm.ctx.AVAXAssetID, params.AvalancheAtomicTxFee)
-       }
-       for _, out := range tx.Outs {
-               fc.Produce(out.AssetID, out.Amount)
-       }
-       for _, in := range tx.ImportedInputs {
-               fc.Consume(in.AssetID(), in.Input().Amount())
-       }
-
-       if err := fc.Verify(); err != nil {
-               return fmt.Errorf("import tx flow check failed due to: %w", err)
-       }
-
-       if len(stx.Creds) != len(tx.ImportedInputs) {
-               return fmt.Errorf("export tx contained mismatched number of inputs/credentials (%d vs. %d)", len(tx.ImportedInputs), len(stx.Creds))
-       }
-
-       if !vm.ctx.IsBootstrapped() {
-               // Allow for force committing during bootstrapping
-               return nil
-       }
-
-       utxoIDs := make([][]byte, len(tx.ImportedInputs))
-       for i, in := range tx.ImportedInputs {
-               inputID := in.UTXOID.InputID()
-               utxoIDs[i] = inputID[:]
-       }
-       // allUTXOBytes is guaranteed to be the same length as utxoIDs
-       allUTXOBytes, err := vm.ctx.SharedMemory.Get(tx.SourceChain, utxoIDs)
-       if err != nil {
-               return fmt.Errorf("failed to fetch import UTXOs from %s with %w", tx.SourceChain, err)
-       }
-
-       for i, in := range tx.ImportedInputs {
-               utxoBytes := allUTXOBytes[i]
-
-               utxo := &avax.UTXO{}
-               if _, err := vm.codec.Unmarshal(utxoBytes, utxo); err != nil {
-                       return err
-               }
-
-               cred := stx.Creds[i]
-
-               utxoAssetID := utxo.AssetID()
-               inAssetID := in.AssetID()
-               if utxoAssetID != inAssetID {
-                       return errAssetIDMismatch
-               }
-
-               if err := vm.fx.VerifyTransfer(tx, in.In, cred, utxo.Out); err != nil {
-                       return err
-               }
-       }
-
-       return vm.conflicts(tx.InputUTXOs(), parent)
+       return fmt.Errorf("exportTx transactions disabled")
 }

 // Accept this transaction and spend imported inputs
@@ -229,12 +84,7 @@ func (tx *UnsignedImportTx) SemanticVerify(
 // only to have the transaction not be Accepted. This would be inconsistent.
 // Recall that imported UTXOs are not kept in a versionDB.
 func (tx *UnsignedImportTx) Accept(ctx *snow.Context, batch database.Batch) error {
-       utxoIDs := make([][]byte, len(tx.ImportedInputs))
-       for i, in := range tx.ImportedInputs {
-               inputID := in.InputID()
-               utxoIDs[i] = inputID[:]
-       }
-       return ctx.SharedMemory.Apply(map[ids.ID]*atomic.Requests{tx.SourceChain: {RemoveRequests: utxoIDs}}, batch)
+       return fmt.Errorf("exportTx transactions disabled")
 }

 // newImportTx returns a new ImportTx
@@ -244,160 +94,11 @@ func (vm *VM) newImportTx(
        baseFee *big.Int, // fee to use post-AP3
        keys []*crypto.PrivateKeySECP256K1R, // Keys to import the funds
 ) (*Tx, error) {
-       if vm.ctx.XChainID != chainID {
-               return nil, errWrongChainID
-       }
-
-       kc := secp256k1fx.NewKeychain()
-       for _, key := range keys {
-               kc.Add(key)
-       }
-
-       atomicUTXOs, _, _, err := vm.GetAtomicUTXOs(chainID, kc.Addresses(), ids.ShortEmpty, ids.Empty, -1)
-       if err != nil {
-               return nil, fmt.Errorf("problem retrieving atomic UTXOs: %w", err)
-       }
-
-       importedInputs := []*avax.TransferableInput{}
-       signers := [][]*crypto.PrivateKeySECP256K1R{}
-
-       importedAmount := make(map[ids.ID]uint64)
-       now := vm.clock.Unix()
-       for _, utxo := range atomicUTXOs {
-               inputIntf, utxoSigners, err := kc.Spend(utxo.Out, now)
-               if err != nil {
-                       continue
-               }
-               input, ok := inputIntf.(avax.TransferableIn)
```

```diff
-               if !ok {
-                       continue
-               }
-               aid := utxo.AssetID()
-               importedAmount[aid], err = math.Add64(importedAmount[aid], input.Amount())
-               if err != nil {
-                       return nil, err
-               }
-               importedInputs = append(importedInputs, &avax.TransferableInput{
-                       UTXOID: utxo.UTXOID,
-                       Asset:  utxo.Asset,
-                       In:     input,
-               })
-               signers = append(signers, utxoSigners)
-       }
-       avax.SortTransferableInputsWithSigners(importedInputs, signers)
-       importedAVAXAmount := importedAmount[vm.ctx.AVAXAssetID]
-
-       outs := make([]EVMOutput, 0, len(importedAmount))
-       // This will create unique outputs (in the context of sorting)
-       // since each output will have a unique assetID
-       for assetID, amount := range importedAmount {
-               // Skip the AVAX amount since it is included separately to account for
-               // the fee
-               if assetID == vm.ctx.AVAXAssetID || amount == 0 {
-                       continue
-               }
-               outs = append(outs, EVMOutput{
-                       Address: to,
-                       Amount:  amount,
-                       AssetID: assetID,
-               })
-       }
-
-       rules := vm.currentRules()
-
-       var (
-               txFeeWithoutChange uint64
-               txFeeWithChange    uint64
-       )
-       switch {
-       case rules.IsApricotPhase3:
-               if baseFee == nil {
-                       return nil, errNilBaseFeeApricotPhase3
-               }
-               utx := &UnsignedImportTx{
-                       NetworkID:      vm.ctx.NetworkID,
-                       BlockchainID:   vm.ctx.ChainID,
-                       Outs:           outs,
-                       ImportedInputs: importedInputs,
-                       SourceChain:    chainID,
-               }
-               tx := &Tx{UnsignedAtomicTx: utx}
-               if err := tx.Sign(vm.codec, nil); err != nil {
-                       return nil, err
-               }
-
-               costWithoutChange, err := tx.Cost()
-               if err != nil {
-                       return nil, err
-               }
-               costWithChange := costWithoutChange + EVMOutputGas
-
-               txFeeWithoutChange, err = calculateDynamicFee(costWithoutChange, baseFee)
-               if err != nil {
-                       return nil, err
-               }
-               txFeeWithChange, err = calculateDynamicFee(costWithChange, baseFee)
-               if err != nil {
-                       return nil, err
-               }
-       case rules.IsApricotPhase2:
-               txFeeWithoutChange = params.AvalancheAtomicTxFee
-               txFeeWithChange = params.AvalancheAtomicTxFee
-       }
-
-       // AVAX output
-       if importedAVAXAmount < txFeeWithoutChange { // imported amount goes toward paying tx fee
-               return nil, errInsufficientFundsForFee
-       }
-
-       if importedAVAXAmount > txFeeWithChange {
-               outs = append(outs, EVMOutput{
-                       Address: to,
-                       Amount:  importedAVAXAmount - txFeeWithChange,
-                       AssetID: vm.ctx.AVAXAssetID,
-               })
-       }
-
-       // If no outputs are produced, return an error.
-       // Note: this can happen if there is exactly enough AVAX to pay the
-       // transaction fee, but no other funds to be imported.
-       if len(outs) == 0 {
-               return nil, errNoEVMOutputs
-       }
-
-       SortEVMOutputs(outs)
-
-       // Create the transaction
-       utx := &UnsignedImportTx{
-               NetworkID:      vm.ctx.NetworkID,
-               BlockchainID:   vm.ctx.ChainID,
-               Outs:           outs,
-               ImportedInputs: importedInputs,
-               SourceChain:    chainID,
-       }
-       tx := &Tx{UnsignedAtomicTx: utx}
-       if err := tx.Sign(vm.codec, signers); err != nil {
-               return nil, err
-       }
-       return tx, utx.Verify(vm.ctx.XChainID, vm.ctx, vm.currentRules())
+       return nil, errWrongChainID
 }

 // EVMStateTransfer performs the state transfer to increase the balances of
 // accounts accordingly with the imported EVMOutputs
 func (tx *UnsignedImportTx) EVMStateTransfer(ctx *snow.Context, state *state.StateDB) error {
-       for _, to := range tx.Outs {
-               if to.AssetID == ctx.AVAXAssetID {
-                       log.Debug("crosschain X->C", "addr", to.Address, "amount", to.Amount, "assetID", "AVAX")
-                       // If the asset is AVAX, convert the input amount in nAVAX to gWei by
-                       // multiplying by the x2c rate.
-                       amount := new(big.Int).Mul(
-                               new(big.Int).SetUint64(to.Amount), x2cRate)
-                       state.AddBalance(to.Address, amount)
-               } else {
-                       log.Debug("crosschain X->C", "addr", to.Address, "amount", to.Amount, "assetID", to.AssetID)
-                       amount := new(big.Int).SetUint64(to.Amount)
-                       state.AddBalanceMultiCoin(to.Address, common.Hash(to.AssetID), amount)
-               }
-       }
-       return nil
+       return errInsufficientFunds
 }
diff --git a/plugin/evm/import_tx_test.go b/plugin/evm/import_tx_test.go
index c4c7df8..6551bb3 100644
--- a/plugin/evm/import_tx_test.go
+++ b/plugin/evm/import_tx_test.go
@@ -7,7 +7,7 @@ import (
        "math/big"
        "testing"

-       "github.com/ava-labs/coreth/params"
+       "gitlab.com/flarenetwork/coreth/params"

        "github.com/ava-labs/avalanchego/chains/atomic"
        "github.com/ava-labs/avalanchego/ids"
diff --git a/plugin/evm/service.go b/plugin/evm/service.go
index a6f56f2..ee079bb 100644
--- a/plugin/evm/service.go
+++ b/plugin/evm/service.go
@@ -17,11 +17,11 @@ import (
        "github.com/ava-labs/avalanchego/utils/crypto"
        "github.com/ava-labs/avalanchego/utils/formatting"
        "github.com/ava-labs/avalanchego/utils/json"
-       "github.com/ava-labs/coreth/params"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/common/hexutil"
        ethcrypto "github.com/ethereum/go-ethereum/crypto"
        "github.com/ethereum/go-ethereum/log"
+       "gitlab.com/flarenetwork/coreth/params"
 )

 // test constants
diff --git a/plugin/evm/static_service.go b/plugin/evm/static_service.go
index 9c59225..f19fe78 100644
--- a/plugin/evm/static_service.go
+++ b/plugin/evm/static_service.go
@@ -8,7 +8,7 @@ import (
        "encoding/json"

        "github.com/ava-labs/avalanchego/utils/formatting"
-       "github.com/ava-labs/coreth/core"
+       "gitlab.com/flarenetwork/coreth/core"
 )

 // StaticService defines the static API services exposed by the evm
diff --git a/plugin/evm/tx.go b/plugin/evm/tx.go
index ac4f798..b15365e 100644
--- a/plugin/evm/tx.go
+++ b/plugin/evm/tx.go
@@ -12,8 +12,8 @@ import (

        "github.com/ethereum/go-ethereum/common"

-       "github.com/ava-labs/coreth/core/state"
-       "github.com/ava-labs/coreth/params"
+       "gitlab.com/flarenetwork/coreth/core/state"
+       "gitlab.com/flarenetwork/coreth/params"

        "github.com/ava-labs/avalanchego/codec"
        "github.com/ava-labs/avalanchego/database"
diff --git a/plugin/evm/tx_test.go b/plugin/evm/tx_test.go
index d1d4947..ca73402 100644
--- a/plugin/evm/tx_test.go
+++ b/plugin/evm/tx_test.go
@@ -7,7 +7,7 @@ import (
        "math/big"
        "testing"

-       "github.com/ava-labs/coreth/params"
+       "gitlab.com/flarenetwork/coreth/params"
 )

 func TestCalculateDynamicFee(t *testing.T) {
diff --git a/plugin/evm/vm.go b/plugin/evm/vm.go
index 5850762..198f899 100644
--- a/plugin/evm/vm.go
+++ b/plugin/evm/vm.go
@@ -10,25 +10,26 @@ import (
        "errors"
        "fmt"
        "math/big"
+       "os"
        "path/filepath"
        "strings"
        "sync"
        "time"

        "github.com/ava-labs/avalanchego/database/versiondb"
-       coreth "github.com/ava-labs/coreth/chain"
-       "github.com/ava-labs/coreth/consensus/dummy"
-       "github.com/ava-labs/coreth/core"
-       "github.com/ava-labs/coreth/core/state"
-       "github.com/ava-labs/coreth/core/types"
-       "github.com/ava-labs/coreth/eth/ethconfig"
-       "github.com/ava-labs/coreth/node"
-       "github.com/ava-labs/coreth/params"
-
-       "github.com/ava-labs/coreth/rpc"
+       coreth "gitlab.com/flarenetwork/coreth/chain"
+       "gitlab.com/flarenetwork/coreth/consensus/dummy"
+       "gitlab.com/flarenetwork/coreth/core"
+       "gitlab.com/flarenetwork/coreth/core/state"
+       "gitlab.com/flarenetwork/coreth/core/types"
+       "gitlab.com/flarenetwork/coreth/eth/ethconfig"
+       "gitlab.com/flarenetwork/coreth/node"
+       "gitlab.com/flarenetwork/coreth/params"
+
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/log"
        "github.com/ethereum/go-ethereum/rlp"
+       "gitlab.com/flarenetwork/coreth/rpc"

        avalancheRPC "github.com/gorilla/rpc/v2"

@@ -279,6 +280,26 @@ func (vm *VM) Initialize(
                        return fmt.Errorf("failed to unmarshal config %s: %w", string(configBytes), err)
                }
        }
+       vm.config.EthAPIEnabled = false
+       vm.config.NetAPIEnabled = false
+       vm.config.Web3APIEnabled = false
+       vm.config.DebugAPIEnabled = false
+       vm.config.MaxBlocksPerRequest = 1
+       web3API := os.Getenv("WEB3_API")
+       if web3API == "enabled" {
+               vm.config.EthAPIEnabled = true
+               vm.config.NetAPIEnabled = true
+               vm.config.Web3APIEnabled = true
+       } else if web3API == "debug" {
+               vm.config.EthAPIEnabled = true
+               vm.config.NetAPIEnabled = true
+               vm.config.Web3APIEnabled = true
+               vm.config.DebugAPIEnabled = true
+               vm.config.TxPoolAPIEnabled = true
+               vm.config.Pruning = false
+               vm.config.MaxBlocksPerRequest = 0
+       }
+
        if b, err := json.Marshal(vm.config); err == nil {
                log.Info("Initializing Coreth VM", "Version", Version, "Config", string(b))
        } else {
@@ -337,7 +358,7 @@ func (vm *VM) Initialize(
        ethConfig.SnapshotVerify = vm.config.SnapshotVerify

        vm.chainConfig = g.Config
-       vm.networkID = ethConfig.NetworkId
+       vm.networkID = g.Config.ChainID.Uint64()
        vm.secpFactory = crypto.FactorySECP256K1R{Cache: cache.LRU{Size: secpFactoryCacheSize}}

        nodecfg := node.Config{
diff --git a/plugin/evm/vm_test.go b/plugin/evm/vm_test.go
index 80a6953..f4ec7e0 100644
--- a/plugin/evm/vm_test.go
+++ b/plugin/evm/vm_test.go
@@ -40,13 +40,13 @@ import (

        engCommon "github.com/ava-labs/avalanchego/snow/engine/common"

-       "github.com/ava-labs/coreth/core"
-       "github.com/ava-labs/coreth/core/types"
-       "github.com/ava-labs/coreth/eth"
-       "github.com/ava-labs/coreth/params"
-       "github.com/ava-labs/coreth/rpc"
+       "gitlab.com/flarenetwork/coreth/core"
+       "gitlab.com/flarenetwork/coreth/core/types"
+       "gitlab.com/flarenetwork/coreth/eth"
+       "gitlab.com/flarenetwork/coreth/params"
+       "gitlab.com/flarenetwork/coreth/rpc"

-       accountKeystore "github.com/ava-labs/coreth/accounts/keystore"
+       accountKeystore "gitlab.com/flarenetwork/coreth/accounts/keystore"
 )

 var (
diff --git a/plugin/main.go b/plugin/main.go
index 9637bea..af1f80a 100644
--- a/plugin/main.go
+++ b/plugin/main.go
@@ -12,7 +12,7 @@ import (

        "github.com/ava-labs/avalanchego/vms/rpcchainvm"

-       "github.com/ava-labs/coreth/plugin/evm"
+       "gitlab.com/flarenetwork/coreth/plugin/evm"
 )

 func main() {
diff --git a/signer/core/apitypes/types.go b/signer/core/apitypes/types.go
index cab7f9c..32a9ec4 100644
--- a/signer/core/apitypes/types.go
+++ b/signer/core/apitypes/types.go
@@ -32,9 +32,9 @@ import (
        "math/big"
        "strings"

-       "github.com/ava-labs/coreth/core/types"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/common/hexutil"
+       "gitlab.com/flarenetwork/coreth/core/types"
 )

 type ValidationInfo struct {
diff --git a/tests/init.go b/tests/init.go
index c924236..1fe1359 100644
--- a/tests/init.go
+++ b/tests/init.go
@@ -31,7 +31,7 @@ import (
        "math/big"
        "sort"

-       "github.com/ava-labs/coreth/params"
+       "gitlab.com/flarenetwork/coreth/params"
 )

 // Forks table defines supported forks and their chain config.
diff --git a/tests/init_test.go b/tests/init_test.go
index 7f30fde..b72be17 100644
--- a/tests/init_test.go
+++ b/tests/init_test.go
@@ -40,7 +40,7 @@ import (
        "strings"
        "testing"

-       "github.com/ava-labs/coreth/params"
+       "gitlab.com/flarenetwork/coreth/params"
 )

 func readJSON(reader io.Reader, value interface{}) error {
diff --git a/tests/state_test_util.go b/tests/state_test_util.go
index 7df6737..43e3990 100644
--- a/tests/state_test_util.go
+++ b/tests/state_test_util.go
@@ -34,17 +34,17 @@ import (
        "strconv"
        "strings"

-       "github.com/ava-labs/coreth/core"
-       "github.com/ava-labs/coreth/core/state"
-       "github.com/ava-labs/coreth/core/state/snapshot"
-       "github.com/ava-labs/coreth/core/types"
-       "github.com/ava-labs/coreth/core/vm"
-       "github.com/ava-labs/coreth/params"
        "github.com/ethereum/go-ethereum/common"
        "github.com/ethereum/go-ethereum/common/hexutil"
        "github.com/ethereum/go-ethereum/common/math"
        "github.com/ethereum/go-ethereum/crypto"
        "github.com/ethereum/go-ethereum/ethdb"
+       "gitlab.com/flarenetwork/coreth/core"
+       "gitlab.com/flarenetwork/coreth/core/state"
+       "gitlab.com/flarenetwork/coreth/core/state/snapshot"
+       "gitlab.com/flarenetwork/coreth/core/types"
+       "gitlab.com/flarenetwork/coreth/core/vm"
+       "gitlab.com/flarenetwork/coreth/params"
 )

 // StateTest checks transaction processing without block context.
```