Introduction to Smart Contracts and Solidity

Part 6 - The Ethereum Virtual Machine Message Calls

The EVM - Message Calls

Mechanism for inter-contract communication

- Call to other contracts
- Sending Ethereum to non-contract accounts

Similarity with transactions

- Each transaction is wrapped in a message call

Each new contract call

- New memory instance
- Stack frame with call arguments + call-related data

Message call - parameters

Message parameters

- Source
- Target
- Data payload (input data)
- Ether
- Gas
- Return data

Message call - gas and exceptions

Contract and gas allocation

- Sets the amount of gas for inner message calls
- At most 63/64 parts forwarded

Exception bubbling up

- From the inner message call
- To the outer message call

Message call - allocation

Memory instance allocation

- Receives the call payload

Call payload and return

- Function arguments reside in Calldata memory area
- Call contract returns assigned to the caller’s memory
- All message calls are synchronous

Message call - properties

Message call depth

- 1024 levels in theory
- About 1000 levels in practice
- Not recommended for array iteration (via recursion)
- Not recommended for complex operations
- Use loops instead

`callcode` and `delegatecall` and Libraries

Special variants of message calls

`callcode`

- Similar to an ordinary message call
- Code is executed in the context of the calling contract

`delegatecall`

- Identical to `callcode`, but…
- …keeps `msg.sender` and `msg.value` unchanged

Library-like behavior

- Dynamic code loading from remote contracts

Logs

Specially indexed data structure

- Logging functionality
- Event implementation
- Storage in Ethereum blockchain + Bloom filters
- Access from outside the contract
  - Light clients

`create`

Special operation code `create`

- Contract creation

Differs from ordinary message call

- Payload data is treated as code
- Payload data is executed
- Return data upon creation → contract address

`self-destruct` and deactivate

Special operation code `selfdestruct`

- Contract destruction
- Still exists in blockchain history

`selfdestruct` possible origins

- The calling contract
- `callcode` remote contract
- `delegatecall` remote contract

Danger

- Possible future misplacement of Ether

`self-destruct` and deactivate

Better approach

- Contract deactivation (not an operation code, but an approach)
- Disabling of the internal states = passive contract
- All functions revert
- Prevents Ether misplacement

Precompiled Contracts

Special group of contracts

- Reserved contract addresses at locations 1-8
- Behavior and gas consumption not defined by the our code
- Implemented in EVM runtime environment

Future considerations

- Expected precompiled contracts range `1` and `0xffff`
- Possible deviation with EVM-compatible chains
    - Different precompiled contract set