## 스마트 컨트랙트 개발

Quorum은 스마트 컨트랙트 작성 시 표준 [Solidity](https://solidity.readthedocs.io/en/develop/)를 사용하고 있으며, 일반적으로 이더리움에서 스마트 컨트랙트를 개발할 때와 동일한 방법으로 개발하시면 됩니다. 스마트 컨트랙트는 퍼블릭(즉, Quorum 네트워크의 모든 참여자가 볼 수 있고 실행 가능한 형태) 혹은 지정한 참여자들에게만 공개하는 프라이빗한 형태로 사용할 수 있습니다. 유의하실 점은 Quorum은 새로운 컨트랙트 형태를 도입하지 않는다는 점입니다. 대신, [트랜잭션 처리 및 프라이버시] 섹션에서 설명한 것과 같이 퍼블릭과 프라이빗 컨트랙트는 개념적으로만 이루어집니다.

### 퍼블릭 트랜잭션/컨트랙트 생성

Quorum 네트워크 상에서 모든 참여자들이 트랜잭션/스마트 컨트랙트를 조회하거나 실행 가능하게 하기 위해서는 네트워크에 이더리움 트랜잭션을 전송하기만 하면 됩니다. (트랜잭션이 컨트랙트를 생성하길 원한다면 to 파라미터는 비워두세요)

표준 이더리움 API에서 수정된 sendTransaction API에 관한 자세한 내용은 [Quorum API](https://github.com/jpmorganchase/quorum/blob/master/docs/api.md) 페이지에서 확인할 수 있습니다.

참고: Quorum 컨트랙트 생성 시 주요 참고사항들은 하단의 ‘Quorum 컨트랙트 디자인 시 고려사항’ 섹션을 참고해주세요.

### 프라이빗 트랜잭션/컨트랙트 생성

Quorum 네트워크 상에서 일부 참여자들만 트랜잭션/스마트 컨트랙트를 실행할 수 있도록 하고 싶은 경우, 표준 이더리움 트랜잭션에 privateFor 파라미터만 추가하시면 됩니다. privateFor 파라미터에는 트랜잭션이나 컨트랙트를 실행할 수 있는 참여자의 퍼블릭 키 목록을 설정하시면 됩니다

JSON 메시지 예시:

'{"jsonrpc":"2.0","method":"eth\_sendTransaction","params":[{"from": $FROM\_AC, "to": $TO\_AC, "data": $CODEHASH, "privateFor": ["$PUBKEY1,PUBKEY2"]}],"id":$ID}'

표준 이더리움 API에서 수정된 sendTransaction API에 관한 자세한 내용은 [Quorum API](https://github.com/jpmorganchase/quorum/blob/master/docs/api.md) 페이지에서 확인할 수 있습니다.

참고: Quorum 컨트랙트 생성 시 주요 참고사항들은 하단의 ‘Quorum 컨트랙트 디자인 고려사항’ 섹션을 참고해주세요.

### Quorum 컨트랙트 디자인 시 고려사항

1. *프라이빗 컨트랙트는 퍼블릭 컨트랙트를 업데이트 할 수 없습니다.* 이는 모든 참여자들이 프라이빗 컨트랙트를 실행할 수는 없는 상황에서 프라이빗 컨트랙트가 퍼블릭 컨트랙트를 업데이트 할 수 있다면 각각의 참여자들이 제각각 다른 상태의 퍼블릭 컨트랙트를 가지게 되기 때문입니다.
2. *퍼블릭* *컨트랙트를 프라이빗 컨트랙트로 변경할 수 없습니다.* 퍼블릭 컨트랙트를 프라이빗으로 변경해야 한다면 블록체인에서 해당 컨트랙트를 삭제하고 새로운 프라이빗 컨트랙트를 생성해야 합니다.

## 허가형 네트워크 셋업

네트워크 권한은 개별 노드 기동 시, --permissioned 플래그를 통해 설정 가능합니다. 해당 플래그가 추가되면 노드는 *<data-dir>* 폴더의 *permissioned-nodes.json* 파일을 조회합니다:

*permissioned-nodes.json* 파일은 해당 노드가 연결하거나 연결을 허용하는 노드 지정자 목록(enode://remotekey@ip:port)입니다.

--permissioned 플래그를 설정하였는데 *permissioned-nodes.json* 파일이 비어있거나 노드의 *<data-dir>* 폴더에 없다면, 노드가 기동은 하지만 다른 어떤 노드와도 연결할 수 없습니다. 연결을 시도하거나 요청받는 경우 발생하는 오류들은 로그 파일에 기록됩니다.

--permissioned 플래그는 geth 옵션의 기타 리스트에서 확인 가능합니다:

$ geth --help  
 ( truncated output ) MISCELLANEOUS OPTIONS: --permissioned If enabled, the node will allow only a defined list of nodes to connect

참고: 네트워크 접속 허용 권한을 개별 노드에게 위임하는 것은 여러 문제들을 야기할 수 있습니다. 이를 방지하기 위해 권한 설정을 스마트 컨트랙트 기반 모델로 변경하는 것에 대하여 [[제품 로드맵|제품 로드맵]]에서 확인 가능합니다.

### 네트워크 권한 설정(Permissioning) 과정:

1. *<data-dir>* 폴더에 *permissioned-nodes.json* 파일을 생성하세요. JSON 포맷인지 꼭 확인하십시오.
2. 해당 파일에 셋업 과정에서 설정(static-nodes.json) 하거나 연결하고자 하는 Quorum 네트워크 상 노드들의 enode 아이디들을 기술하세요.
3. 노드를기동 시, --permissioned 커맨드라인 플래그를 설정하세요.

*permissioned-nodes.json* 파일의 형식은 다음과 같습니다:

[  
"enode://remoteky1@ip1:port1", "enode://remoteky1@ip2:port2", "enode://remoteky1@ip3:port3", ]

예시 (가독성을 위해 노드 아이디의 끝부분 생략):

[ "enode://8475a01f22a1f48116dc1f0d22ecaaaf77e[::]:30301", "enode://b5660501f496e60e59ded734a889c97b7da[::]:30302", "enode://54bd7ff4bd971fb80493cf4706455395917[::]:30303"]

위 예시는 해당 노드가 이 화이트리스트 상의 3개의 노드로부터 들어오거나 나가는 연결만 수용할 수 있음을 보여줍니다.

### 새로운 노드 추가하기:

*permissioned-nodes.json*에 더해지는 추가사항들은 노드로 들어오거나 나가는 요청이 생성될 때 서버가 즉각적으로 반영합니다. *permissioned-nodes.json* 파일의 변경을 반영하기 위해 노드를 다시 시작할 필요가 없습니다.

### 기존 노드 삭제하기:

*permissioned-nodes.json* 파일에서 기존에 연결되어 있던 노드를 제거해도 즉각적으로 반영되지는 않습니다. 하지만 어떤 이유로든 연결이 끊어지고, 해당 노드 아이디로부터 연결 요청이 들어오면 요청은 반려될 것입니다.

참고: 노드 제거의 즉각적 반영은 [[제품 로드맵|제품 로드맵]]에서 확인 가능합니다.

## Quorum API

자세한 내용은 [Quorum API](https://github.com/jpmorganchase/quorum/blob/master/docs/api.md) 페이지를 참고해주세요.

## ZSL 사용하기

J.P. Morgan과 Zcash 팀은 Quorum에 ZSL(Zcash Security Layer) 적용에 대한 개념 증명 (PoC, Proof of Concept)을 위해 파트너십을 맺었습니다. 이는 ZSL 기반의 퍼블릭 스마트 컨트랙트인 ‘z-contracts’를 이용하여 디지털 자산(z-tokens )을 발행할 수 있도록 합니다. ‘Z-tokens’는 퍼블릭 네트워크에는 노출시키지 않으면서 트랜잭션의 프라이버시를 보장할 수 있습니다. 보호한 트랜잭션이 처리되었는지는 프라이빗 컨트랙트를 통해 확인할 수 있습니다. 프라이빗 컨트랙트의 상태가 퍼블릭 ‘z-contracts’를 이용해 보호된 트랜잭션에 따라 변경되었다면 정상적으로 처리되었다는 것을 의미합니다.

이러한 Constellation과 Tessera의 프라이빗 컨트랙트와 ZSL의 ‘z-contracts’ 조합은 완전한 프라이버시와 비밀을 보장하는 동시에 프라이빗 컨트랙트의 역할을 ‘z-tokens’의 트랜잭션 보호 기능을 통해 해결할 수 있도록 합니다.

더 많은 정보는 wiki의 [|ZSL]] 페이지에서 확인하세요.

## 네트워크와 체인 아이디(Chain ID)

초기 이더리움 네트워크는 네트워크 아이디(Network ID)를 사용하였으나 [EIP-155](https://github.com/ethereum/EIPs/blob/master/EIPS/eip-155.md)이후에는 체인 아이디(Chain ID)를 통해 운영하고 있습니다.

EIP-155 이전에 ‘네트워크 아이디’와 ‘체인 아이디’는 비슷한 의미로 혼용하여 사용하였으나, 이후에는 서로 다른 의미를 갖게 되었습니다.

네트워크 아이디는 피어(peer)가 운영하고 있는 체인이 아닌 피어의 속성입니다. 네트워크 아이디는 --networkid <id> 커맨드 라인을 통해 전달할 수 있습니다. 네트워크 아이디는 서로 다른 네트워크 아이디로 운영하고 있는 피어들을 분리하기 위해 사용됨으로, 서로 다른 네트워크 아이디를 가지는 노드 간의 동기화는 불가능합니다. 하지만, 네트워크 아이디 변경을 통한 피어간 분리는 Quorum의 --permissioned 플래그에 비해 보안성이 낮기 때문에 단순한 분리를 위해서만 사용됩니다.

체인 아이디는 노드에 의해 운영되는 체인의 속성으로 트랜잭션의 리플레이 방지(replay protection)를 위해 사용됩니다. 이는 EIP-155 이전, 트랜잭션이 서명되어 있었다면 특정 체인에서 다른 체인으로 트랜잭션을 복사하여 실행하는 리플레이 공격(replay attack)을 방지하기 위함입니다.

체인 아이디를 설정하는 것은 트랜잭션의 파라미터들 중 하나를 바꾸게 되고 이를 V 파라미터(V parameter)라고 합니다. EIP에서 설명하듯이, V 파라미터는 ‘2\*ChainID + 35/36’로 설정됩니다. 이더리움 재단의 메인넷은 체인 아이디 1을 가지고 있으며, 이는 모든 트랜잭션이 37 혹은 38의 값을 가지고 있다는 것을 의미합니다.

genesis 구성 파일의 config 섹션에 설정된 체인 아이디는 블록 번호가 eip155Block에 설정한 값보다 높을 때만 사용됩니다. 해당 예시는 [quorum-examples genesis files](https://github.com/jpmorganchase/quorum-examples/blob/master/examples/7nodes/genesis.json) 에서 확인할 수 있습니다. 체인의 블록 번호가 eip155Block 번호보다 낮은 상태이면 필요에 따라 얼마든지 변경이 가능하며, 변경 후 geth init 를 재실행하면 됩니다. 이 경우 현재 동기화 프로세스나 저장된 블록을 삭제하거나 수정하지 않습니다!

Quorum에서 트랜잭션의 V 파라미터가 37이나 38로 설정된 경우 프라이빗 트랜잭션으로 간주하기 때문에 체인 아이디 1을 가진 네트워크와 충돌합니다. 이러한 이유로 Quorum은 체인 아이디를 사용하지 않으며, 2.1.0 버전 이후부터는 체인 아이디를 설정하여 노드를 기동하는 경우 실행 즉각 중단될 것입니다. 2.1.0 이전 버전으로 실행하고 있다면 EIP-155 서명이 사용되지 않았기 때문에 체인 아이디 1설정이 가능합니다. 현재 업데이트된 버전을 이용하기 전에 위 사항을 고려하여 genesis 파일을 수정하시고 geth init 를 실행해야 할 것입니다.

## 구성 가능한 트랜잭션 크기:

Quorum은 제네시스 블록을 통해 블록 생성자(operator) 들이 블록에 수용하는 트랜잭션의 최대 사이즈를 늘릴 수 있도록 합니다. 현재 기본으로 설정된 Quorum의 트랜잭션 사이즈는 이더리움의 트랜잭션 사이즈인 32kb에서 64kb로 늘어난 상태이고, 옵션 설정을 통해서는 128kb까지 증가시킬 수 있습니다. 이러한 옵션 구성을 위해서는 genesis 파일의 config 섹션에 다음과 같이 txnSizeLimit을 추가하면 됩니다.   
 "config": { "chainId": 10, "isQuorum":true. ... "txnSizeLimit": 128 }