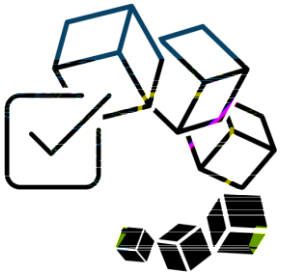
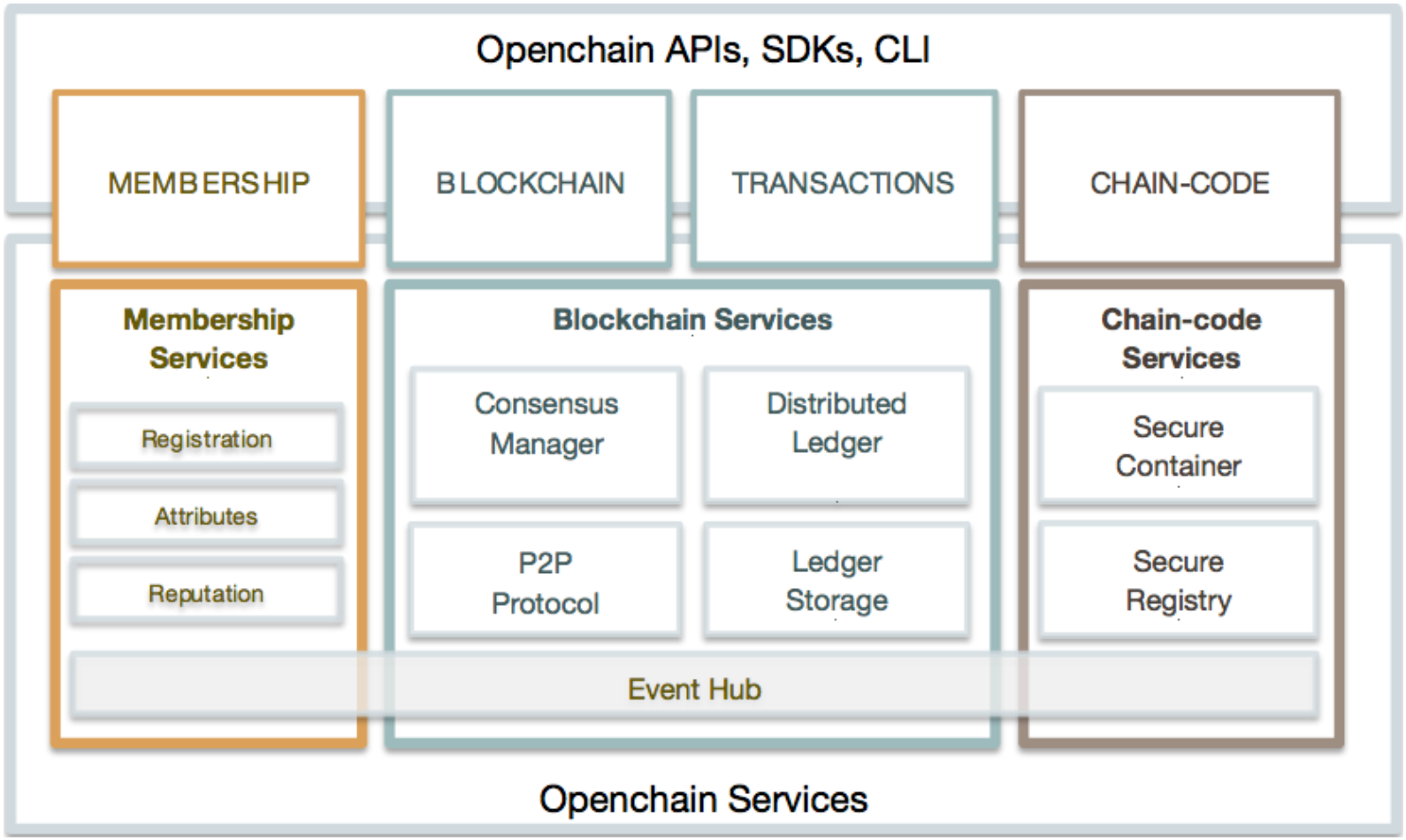
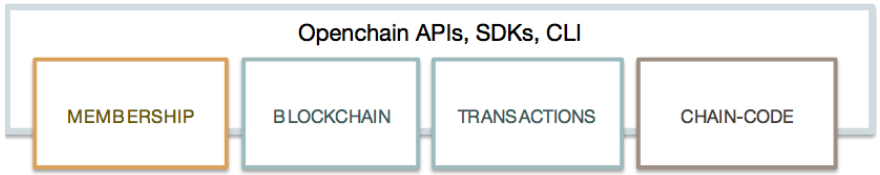


Hyperledger Fabric architecture

Hyperledger Fabric reference architecture



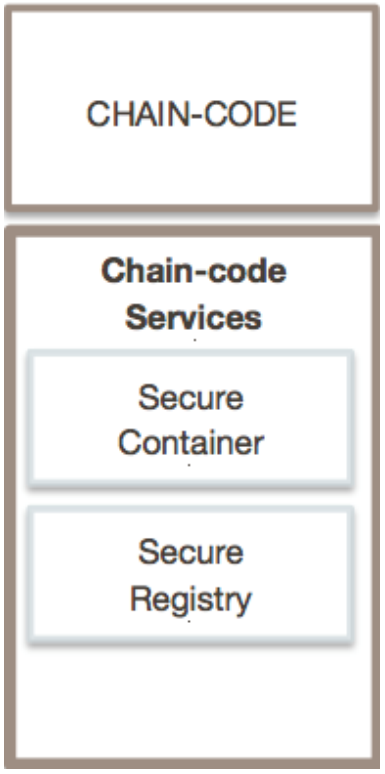
Application programming interface (API)



- Fabric project includes:
  - All common APIs
    - REST API
  - CLI
    - Command Line Interface
  - Software Development Kit
    - SDK

Chaincode

- **Contains Smart Contract** and runs on Validating Nodes
- Stored in secure, machine-independent container
- **Secure Registry** ensures appropriate participant access to chaincode
- Chaincode transactions can execute only within a defined time window
- Chaincode functions can call each other, but **privacy is maintained**



Implementation of chaincode

- **Chaincode** has three important functions:
  - Init()
  - Invoke()
  - Query()
- All three functions take as input a function name and an array of string arguments.

Example of typical business logic in chaincode (from car leasing):

```
if      v.Status        == STATE_TEMPLATE           &&
        current_owner   == caller_name              &&
        caller_role     == ROLE_AUTHORITY           &&
        recipient_role  == ROLE_MANUFACTURER        &&
        v.Scrapped      == false                            {
                             v.Owner  = string(recipient_ecert)
                             v.Status = STATE_MANUFACTURE
} else {
        return nil, errors.New("Permission Denied")
}
```

Application model

- Industry standard application model
- **View Logic** gives mobile or web access into Control Logic
- **Control Logic** connects user interface with Data Model and API to drive transactions
- **Data Model** manages off-chain data including large documents
- **Blockchain Logic** connects:
  - Control Logic to chaincode
  - Data Model to transactions