corda

CORDA CHEAT SHEET

Useful links:

Documentation: docs.corda.net
Slack: slack.corda.net
Forum: discourse.corda.net

RUNNING CORDA

a. Set up your dev environment

https://docs.corda.net/getting-set-up.html

b. Clone the template app

`git clone https://github.com/corda/cordapp-template`

c. Check out the latest milestone (e.g. M12)

`cd cordapp-template && git checkout release-M12`

d. Deploy the nodes

`./gradlew clean deployNodes`

e. Run the nodes

`sh kotlin-source/build/nodes/runnodes`

STATES

ContractState

The base class for on-ledger states

.contract

The `Contract` governing this state's evolution

.participants

The parties able to consume this state

LinearState (extends ContractState)

State representing a 'shared fact' evolving over time

.linearId

An ID shared by all evolutions of the 'shared fact'

.isRelevant(Set<PublicKey>)

Should our vault track this state?

OwnableState (extends ContractState)

State representing fungible assets (cash, oil…)

.owner

The state's current owner

CONTRACTS

Contract

Establishes whether a transaction is valid

.verify(TransactionForContract)

Throws an exception if the transaction is invalid

.legalContractReference

A hash of the contract's legal prose

TRANSACTIONS

TransactionBuilder

A mutable container for building a general transaction

.withItems(vararg Any)

Adds items (states, commands…) to the builder

ServiceHub.signInitialTransaction(TransactionBuilder)

Converts the builder to a signed transaction

TRANSACTIONS (CONT.)

SignedTransaction

A wire transaction, plus associated digital signatures

.verifySignatures(vararg CompositeKey)

Verify the transaction's signatures

.tx.toLedgerTransaction(ServiceHub).verify()

Verify the transaction

ServiceHub.addSignature(SignedTransaction)

Add a digital signature to the transaction

FLOWS

FlowLogic

The actions executed by one side of a flow

.call()

Defines the flow-logic's actions

.send(Party, Any)/receive(Party)/sendAndReceive(Party, Any)

Sends data to/receives data from the specified counterparty

.subFlow(FlowLogic<R>, Boolean)

Invokes a sub-flow that may return a result

.serviceHub

Provides access to the node's services

FLOW ANNOTATIONS

@InitiatingFlow

A flow that is started directly

@InitiatedBy(KClass)

A flow that is only started by a message from an `InitiatingFlow`

@StartableByRPC

Allows the flow to be started via RPC by the node's owner

SERVICE HUB

.networkMapCache

Provides info on other nodes on the network (e.g. notaries…)

.vaultService

Stores the node's current and historic states

.storageService

Stores additional info such as transactions and attachments

.keyManagementService

Manages the node's digital signing keys

.myInfo

Other information about the node

.clock

Provides access to the node's internal time and date

.schedulerService

PROVIDING AN API

a. Subclass CordaPluginRegistry

`class MyPlugin : CordaPluginRegistry() {…}`

b. Override webApis

`override val webApis = listOf(Function(::MyApi))`

c. Register the fully qualified class name of the plugin

…under src/main/resources/META-INF.services