Fabric Deep Dive

Fabric Chaincode and Client SDK

Jingxiao Gu

IBM Research

** News **

Fabric v1.1.0-alpha has been released! 🎉

Online Doc: http://hyperledger-fabric.readthedocs.io/en/v1.1.0-alpha/

Issue Report: https://jira.hyperledger.org

Code Repository: https://gerrit.hyperledger.org

Rocket Chat: https://chat.hyperledger.org

Outline

- Components & Services
- Chaincode
  - Interfaces
  - Instantiation (Launch)
  - Query/Invocation
- Client SDK
- Fabric E2E Live Demo

Preknowledge

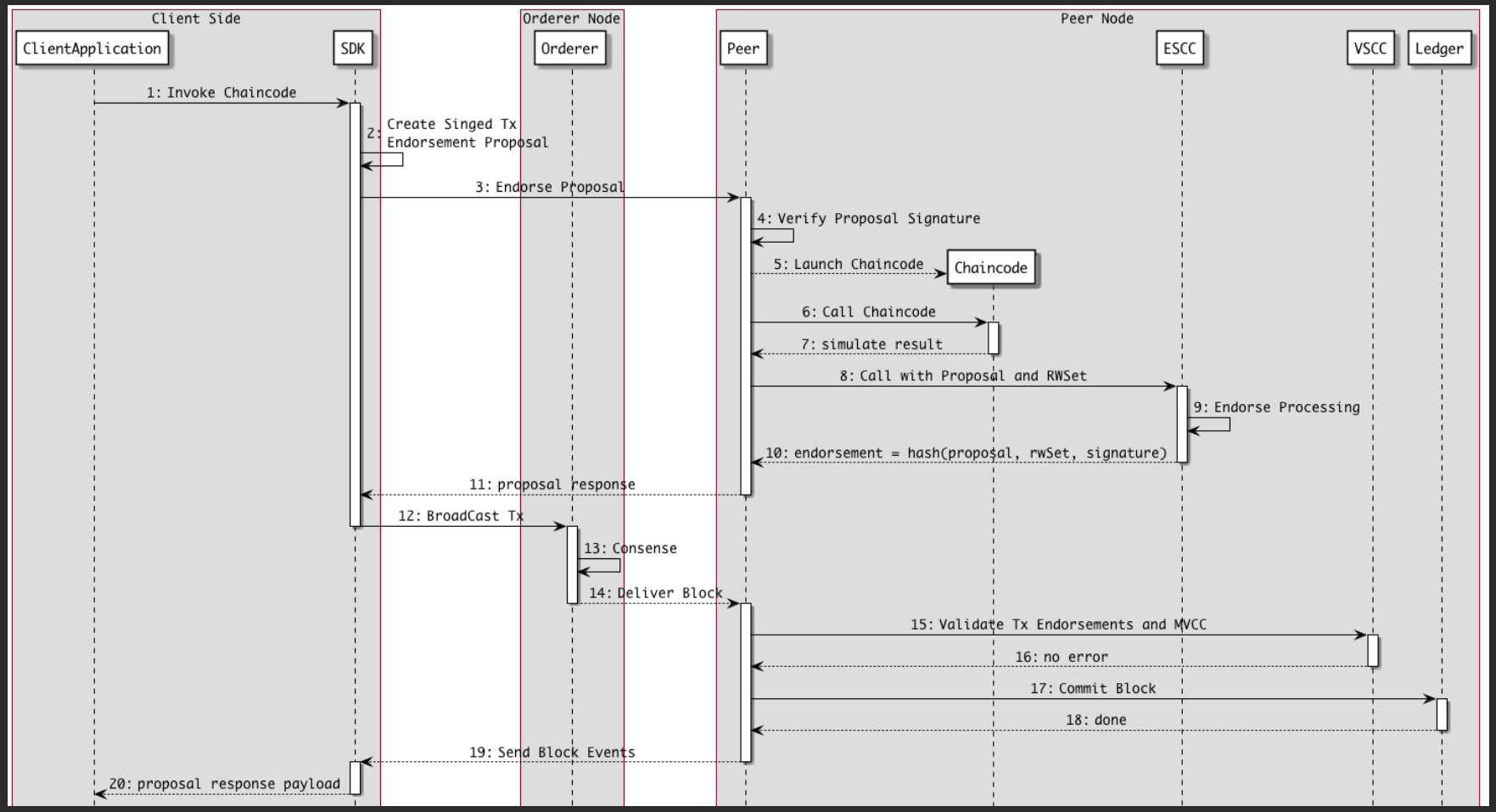
- Programming Language
  - Golang
  - Java
- gRPC Framework
- OOP & UML

Outline

- Components & Services
- Chaincode
  - Interfaces
  - Instantiation (Launch)
  - Query/Invocation
- Client SDK
- Fabric E2E Live Demo

Components & Services

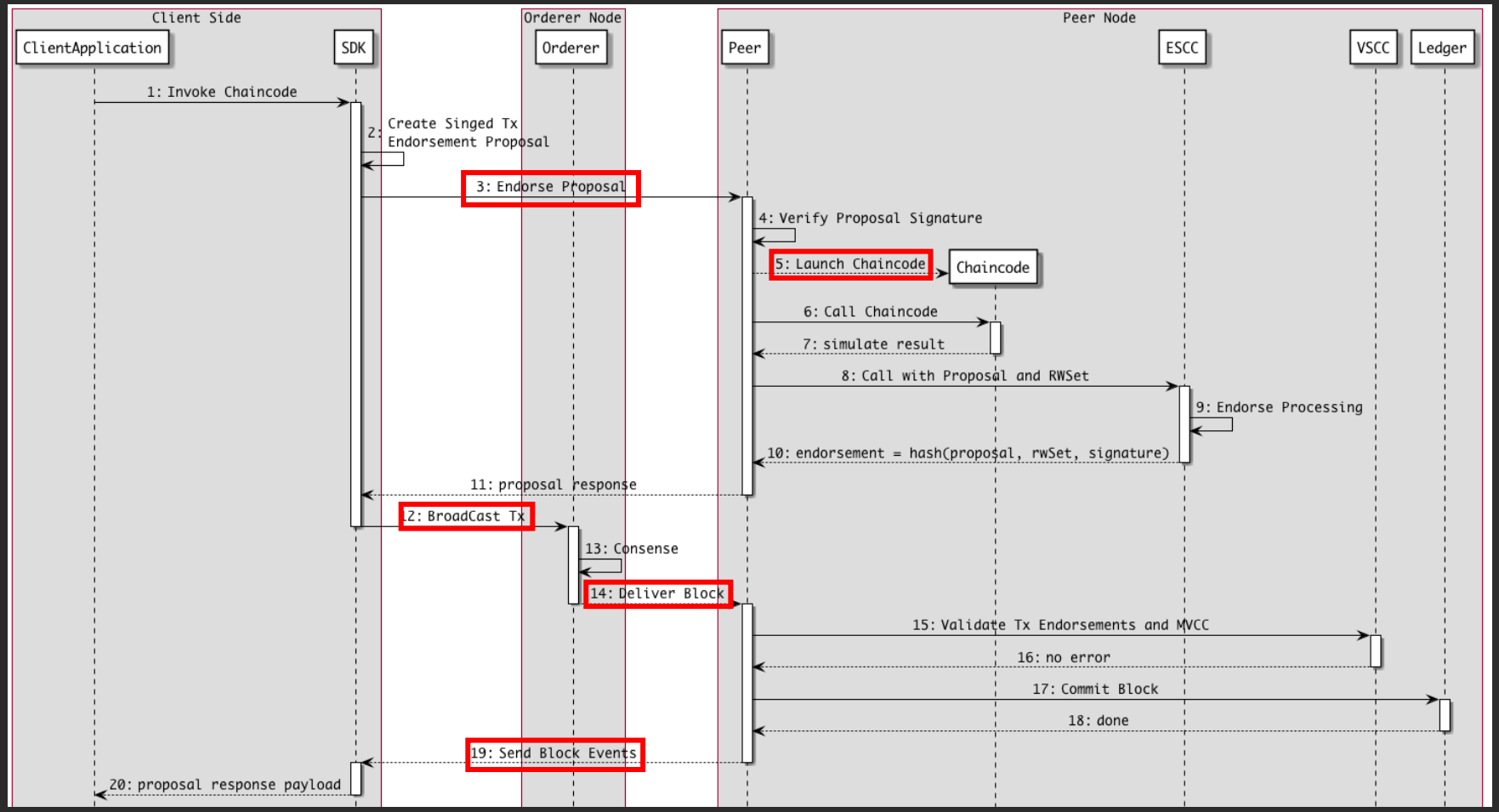
- Overview



Components & Services

- Overview

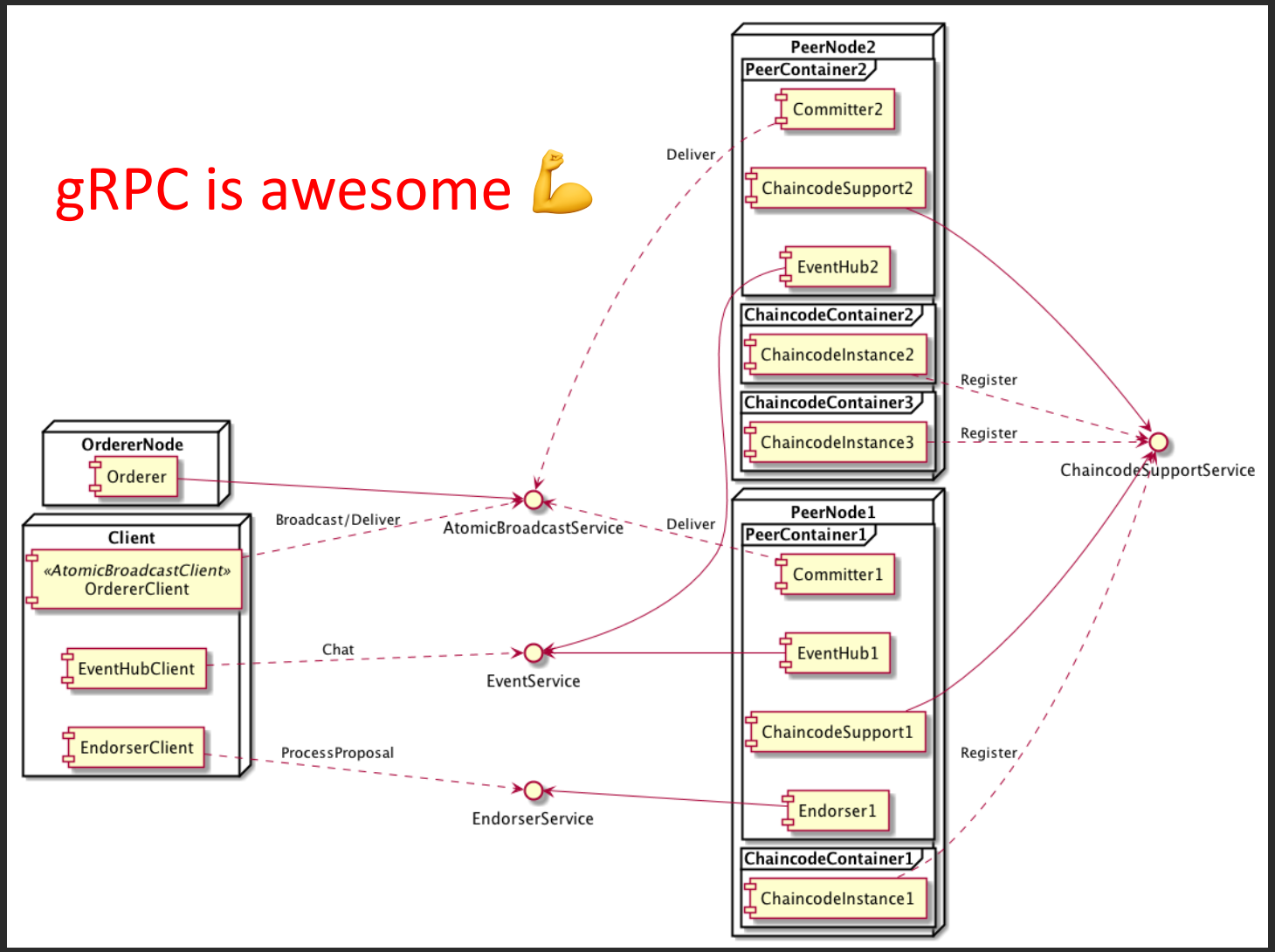
Inter-container/node Invocation



Components & Services

- Inter-container/node Invocations
  - Client → Peer: EndorseProposal
  - Peer ↔ Chaincode: ChaincodeInteractions
  - Client → Orderer: BroadcastTxResponse
  - Orderer → Peer: DeliverBlock
  - Peer → Client: SendBlockEvents

Components & Services



Outline

- Components & Services
- Chaincode
  - Interfaces
  - Instantiation (Launch)
  - Query/Invocation
- Client SDK
- Fabric E2E Live Demo

Chaincode Interfaces

- Typical Golang Chaincode

```go
package main

import (
    "github.com/hyperledger/fabric/core/chaincode/shim"
    pb "github.com/hyperledger/fabric/protos/peer"
)

var logger = shim.NewLogger( name: "demo-chaincode")

type DemoChaincode struct {
}

func (cc *DemoChaincode) Init(stub shim.ChaincodeStubInterface) pb.Response {
    return shim.Success( payload: nil)
}

func (cc *DemoChaincode) Invoke(stub shim.ChaincodeStubInterface) pb.Response {
    return shim.Success( payload: nil)
}

func main() {
    err := shim.Start(new(DemoChaincode))
    if err != nil {
        logger.Error( args: "error starting chaincode: ", err)
    }
}
```

```go
// Chaincode interface must be implemented by all chaincodes. The fabric runs
// the transactions by calling these functions as specified.
type Chaincode interface {
    // Init is called during Instantiate transaction after the chaincode container
    // has been established for the first time, allowing the chaincode to
    // initialize its internal data
    Init(stub ChaincodeStubInterface) pb.Response

    // Invoke is called to update or query the ledger in a proposal transaction.
    // Updated state variables are not committed to the ledger until the
    // transaction is committed.
    Invoke(stub ChaincodeStubInterface) pb.Response
}
```

Chaincode Interfaces

- core/chaincode/shim/interfaces_stable.go

```go
type ChaincodeStubInterface interface {
    GetArgs() [][]byte
    GetStringArgs() []string
    GetFunctionAndParameters() (string, []string)
    GetArgsSlice() ([]byte, error)

    GetTxID() string
    GetChannelID() string
    GetCreator() ([]byte, error)
    GetTransient() (map[string][]byte, error)
    GetBinding() ([]byte, error)
    GetDecorations() map[string][]byte
    GetSignedProposal() (*pb.SignedProposal, error)
    GetTxTimestamp() (*timestamp.Timestamp, error)

    InvokeChaincode(chaincodeName string, args [][]byte, channel string) pb.Response

    GetState(key string) ([]byte, error)
    PutState(key string, value []byte) error
    DelState(key string) error
    GetStateByRange(startKey, endKey string) (StateQueryIteratorInterface, error)
    GetStateByPartialCompositeKey(objectType string, keys []string) (StateQueryIteratorInterface, error)

    CreateCompositeKey(objectType string, attributes []string) (string, error)
    SplitCompositeKey(compositeKey string) (string, []string, error)

    GetQueryResult(query string) (StateQueryIteratorInterface, error)
    GetHistoryForKey(key string) (HistoryQueryIteratorInterface, error)

    SetEvent(name string, payload []byte) error
}
```

Outline

- Components & Services
- Chaincode
  - Interfaces
  - Instantiation (Launch)
  - Query/Invocation
- Client SDK
- Fabric E2E Live Demo

Chaincode Instantiate (Launch)

- Key Steps:
  - **Dev Mode**: ChaincodeSupportService waits for connection from standalone chaincode process
  - **Non-Dev Mode**:
    1. try to start chaincode container (fail for the first time)
    2. generate chaincode building DockerFile for given language
    3. create a Docker container for building chaincode with the DockerFile generated and get chaincode binary
    4. create and start a Docker container for running chaincode
    5. chaincode starts and establishes gRPC connection to ChaincodeSupportService on peer
  - **In Common**: ChaincodeSupportService call `Init` function

Chaincode Instantiate (Launch)

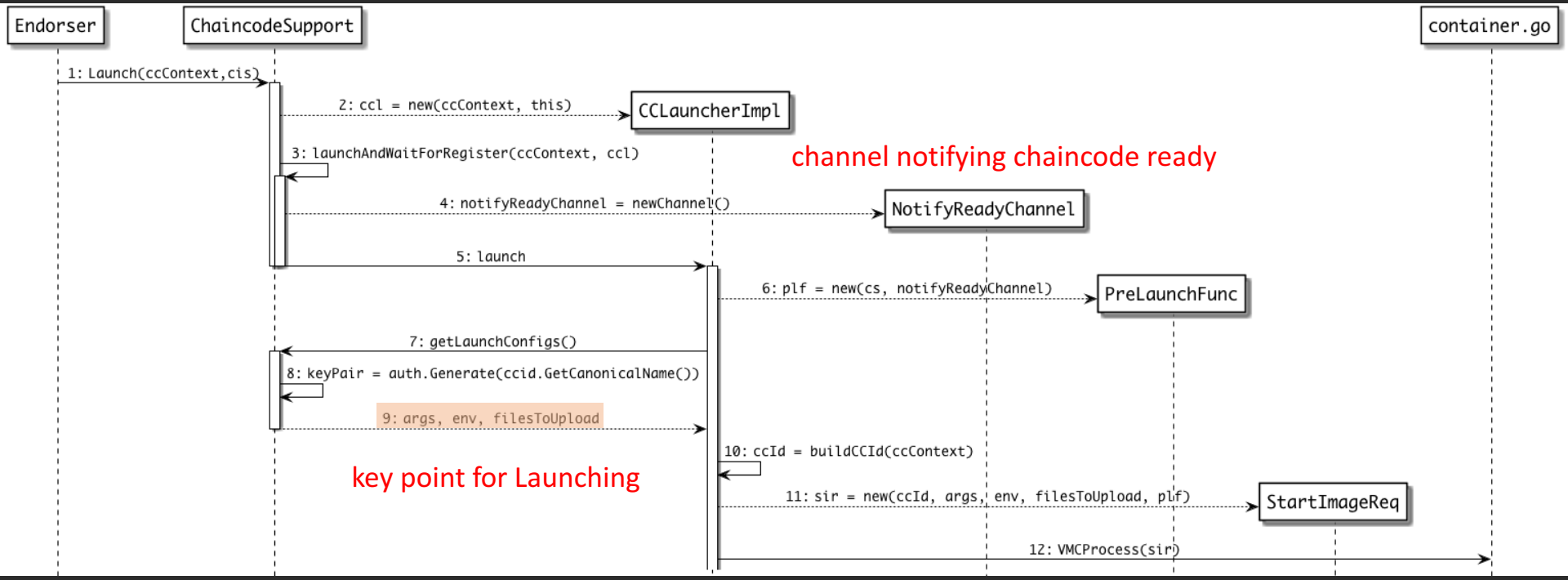
- Sequence Diagrams
  1. Prepare for StartImageReq
  2. Generate chaincode binary
  3. Start chaincode container (peer)
  4. Start chaincode process (inside container)

Chaincode Instantiate (Launch)

- Sequence Diagrams
  1. Prepare for StartImageReq
  2. Generate chaincode binary
  3. Start chaincode container (peer)
  4. Start chaincode process (inside container)

Chaincode Instantiate (Launch)

- Sequence Diagram: Prepare for StartImageReq



Chaincode Instantiate (Launch)

- Sequence Diagram: Prepare for StartImageReq
  - ChaincodeSupport.GetLaunchConfigs()

```
//get args and env given chaincodeID
func (chaincodeSupport *ChaincodeSupport) getLaunchConfigs(cccid *ccprovider.CCContext, cLang pb.ChaincodeSpec_Type)
 (args []string, envs []string, filesToUpload map[string][]byte, err error) {
    canName := cccid.GetCanonicalName()
    envs = []string{"CORE_CHAINCODE_ID_NAME=" + canName}

    var certKeyPair *accesscontrol.CertAndPrivKeyPair
    if chaincodeSupport.peerTLS {
        certKeyPair, err = chaincodeSupport.auth.Generate(cccid.GetCanonicalName())
        if err != nil {
            return  args: nil,  envs: nil,  filesToUpload: nil, errors.WithMessage(err, fmt.Sprintf( format: "failed generating
 TLS cert for %s", cccid.GetCanonicalName()))
        }
        envs = append(envs,  elems: "CORE_PEER_TLS_ENABLED=true")
        envs = append(envs, fmt.Sprintf( format: "CORE_TLS_CLIENT_KEY_PATH=%s", TLSClientKeyPath))
        envs = append(envs, fmt.Sprintf( format: "CORE_TLS_CLIENT_CERT_PATH=%s", TLSClientCertPath))
        envs = append(envs, fmt.Sprintf( format: "CORE_PEER_TLS_ROOTCERT_FILE=%s", TLSClientRootCertPath))
    }
    envs = append(envs, "CORE_CHAINCODE_LOGGING_LEVEL="+chaincodeSupport.chaincodeLogLevel)
    envs = append(envs, "CORE_CHAINCODE_LOGGING_SHIM="+chaincodeSupport.shimLogLevel)
    envs = append(envs, "CORE_CHAINCODE_LOGGING_FORMAT="+chaincodeSupport.logFormat)
```

Chaincode Instantiate (Launch)

- Sequence Diagram: Prepare for StartImageReq
  - ChaincodeSupport.GetLaunchConfigs()

```
    switch cLang {
    case pb.ChaincodeSpec_GOLANG, pb.ChaincodeSpec_CAR:
        args = []string{"chaincode", fmt.Sprintf( format: "-peer.address=%s", chaincodeSupport.peerAddress)}
    case pb.ChaincodeSpec_JAVA:
        args = []string{"java", "-jar", "chaincode.jar", "--peerAddress", chaincodeSupport.peerAddress}
    case pb.ChaincodeSpec_NODE:
        args = []string{"/bin/sh", "-c", fmt.Sprintf( format: "cd /usr/local/src; npm start -- --peer.address %s",
 chaincodeSupport.peerAddress)}

    default:
        return  args: nil,  envs: nil,  filesToUpload: nil, errors.Errorf( format: "unknown chaincodeType: %s", cLang)
    }

    filesToUpload = theChaincodeSupport.getTLSFiles(certKeyPair)

    return args, envs, filesToUpload,  err: nil
}
```

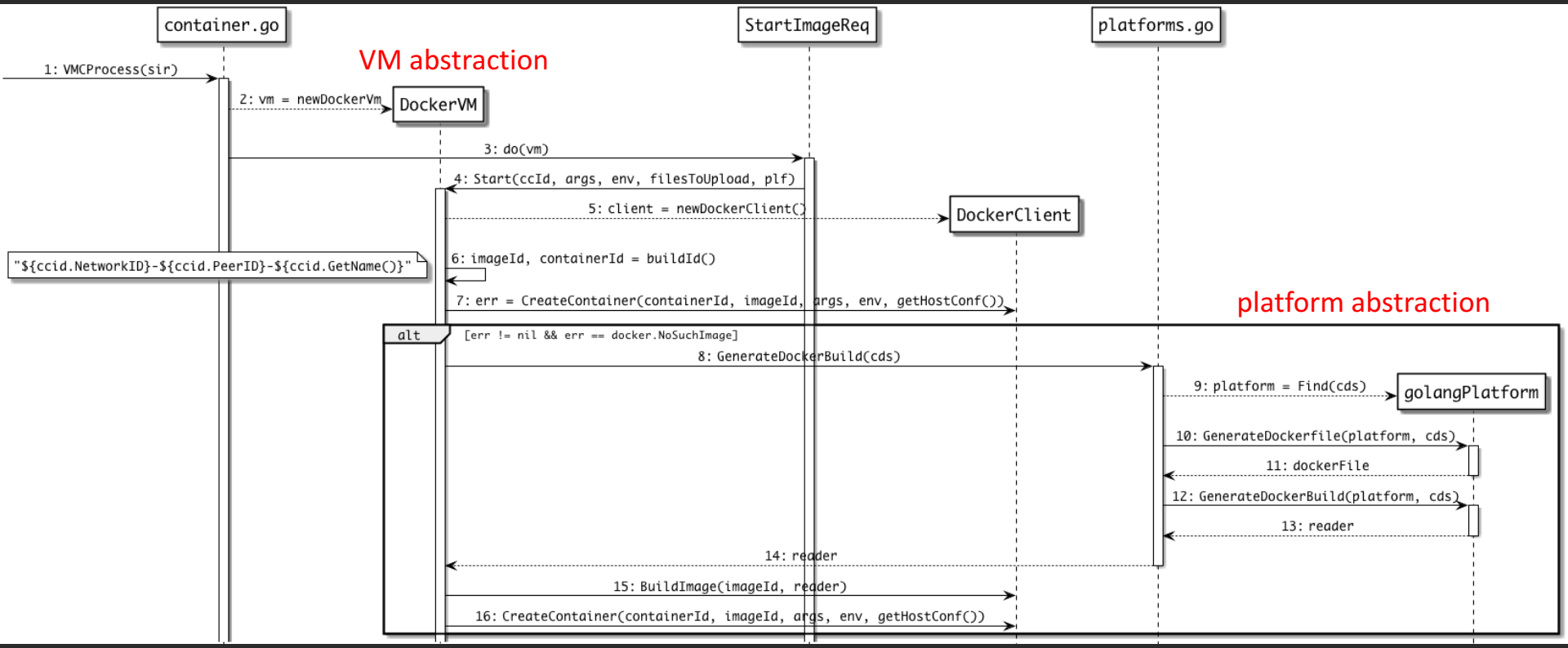
to be refactored 🤔

Chaincode Instantiate (Launch)

- Sequence Diagrams
  1. Prepare for StartImageReq
  2. Generate chaincode binary
  3. Start chaincode container (peer)
  4. Start chaincode process (inside container)

Chaincode Instantiate (Launch)

- Sequence Diagram: Generate chaincode binary



Chaincode Instantiate (Launch)

- Sequence Diagram: Generate chaincode binary
  - Platform abstraction for different languages

```
// Interface for validating the specification and and writing the package for
// the given platform
type Platform interface {
    ValidateSpec(spec *pb.ChaincodeSpec) error
    ValidateDeploymentSpec(spec *pb.ChaincodeDeploymentSpec) error
    GetDeploymentPayload(spec *pb.ChaincodeSpec) ([]byte, error)
    GenerateDockerfile(spec *pb.ChaincodeDeploymentSpec) (string, error)
    GenerateDockerBuild(spec *pb.ChaincodeDeploymentSpec, tw *tar.Writer) error
}
```

  - core/chaincode/platforms/
    - car/platform.go
    - go/platform.go
    - java/platform.go
    - node/platform.go

Chaincode Instantiate (Launch)

- Sequence Diagram: Generate chaincode binary
  - Golang Platform Implementation

```go
// Generates a deployment payload for GOLANG as a series of src/$pkg entries in .tar.gz format
func (goPlatform *Platform) GetDeploymentPayload(spec *pb.ChaincodeSpec) ([]byte, error) {
    /* omit fetching dependencies */
    files := make(Sources, 0)
    payload := bytes.NewBuffer( buf: nil)
    gw := gzip.NewWriter(payload)
    tw := tar.NewWriter(gw)

    for _, file := range files {
        if file.IsMetadata {
            /* omit handling META-INF directory */
        }
        err = cutil.WriteFileToPackage(file.Path, file.Name, tw)
        if err != nil {
            return nil, fmt.Errorf( format: "Error writing %s to tar: %s", file.Name, err)
        }
    }

    tw.Close()
    gw.Close()

    return payload.Bytes(), nil
}
```

Chaincode Instantiate (Launch)

- Sequence Diagram: Generate chaincode binary
  - Golang Platform Implementation

```
func (goPlatform *Platform) GenerateDockerBuild(cds *pb.ChaincodeDeploymentSpec, tw *tar.Writer) error {
    /* omit argument initialization */
    codepackage := bytes.NewReader(cds.CodePackage)
    binpackage := bytes.NewBuffer( buf: nil)
    cmd := fmt.Sprintf( format: "GOPATH=/chaincode/input:$GOPATH go build -tags \"%s\" %s -o /chaincode/output/chaincode %s",
        gotags, ldflagsOpt, pkgname)
    err = util.DockerBuild(util.DockerBuildOptions{
        Cmd:          cmd,
        InputStream:  codepackage,
        OutputStream: binpackage,
        Image: cutil.GetDockerfileFromConfig( path: "chaincode.builder"),
    })
    if err != nil {
        return err
    }

    return cutil.WriteBytesToPackage( name: "binpackage.tar", binpackage.Bytes(), tw)
}
```

core.yaml: chaincode.builder: $(DOCKER_NS)/fabric-ccenv:$(ARCH)-$(PROJECT_VERSION)

Chaincode Instantiate (Launch)

- Sequence Diagram: Generate chaincode binary
  - Golang Platform Implementation

```
func (goPlatform *Platform) GenerateDockerfile(cds *pb.ChaincodeDeploymentSpec) (string, error) {
    var buf []string
    buf = append(buf, "FROM "+cutil.GetDockerfileFromConfig( path: "chaincode.golang.runtime"))
    buf = append(buf,  elems: "ADD binpackage.tar /usr/local/bin")
    return strings.Join(buf,  sep: "\n"), nil
}
```

  - Platform common part

```
func generateDockerfile(platform Platform, cds *pb.ChaincodeDeploymentSpec) ([]byte, error) {
    var buf []string
    base, err := platform.GenerateDockerfile(cds)
    if err != nil {
        return nil, fmt.Errorf( format: "Failed to generate platform-specific Dockerfile: %s", err)
    }
    buf = append(buf, base)
    buf = append(buf, fmt.Sprintf( format: "LABEL %s.chaincode.id.name=\"%s\" \\", metadata.BaseDockerLabel, cds.ChaincodeSpec.ChaincodeId.Name))
    buf = append(buf, fmt.Sprintf( format: "      %s.chaincode.id.version=\"%s\" \\", metadata.BaseDockerLabel, cds.ChaincodeSpec.ChaincodeId.Version))
    buf = append(buf, fmt.Sprintf( format: "      %s.chaincode.type=\"%s\" \\", metadata.BaseDockerLabel, cds.ChaincodeSpec.Type.String()))
    buf = append(buf, fmt.Sprintf( format: "      %s.version=\"%s\" \\", metadata.BaseDockerLabel, metadata.Version))
    buf = append(buf, fmt.Sprintf( format: "      %s.base.version=\"%s\"", metadata.BaseDockerLabel, metadata.BaseVersion))
    buf = append(buf, fmt.Sprintf( format: "ENV CORE_CHAINCODE_BUILDLEVEL=%s", metadata.Version))
    contents := strings.Join(buf,  sep: "\n")
    return []byte(contents), nil
}
```

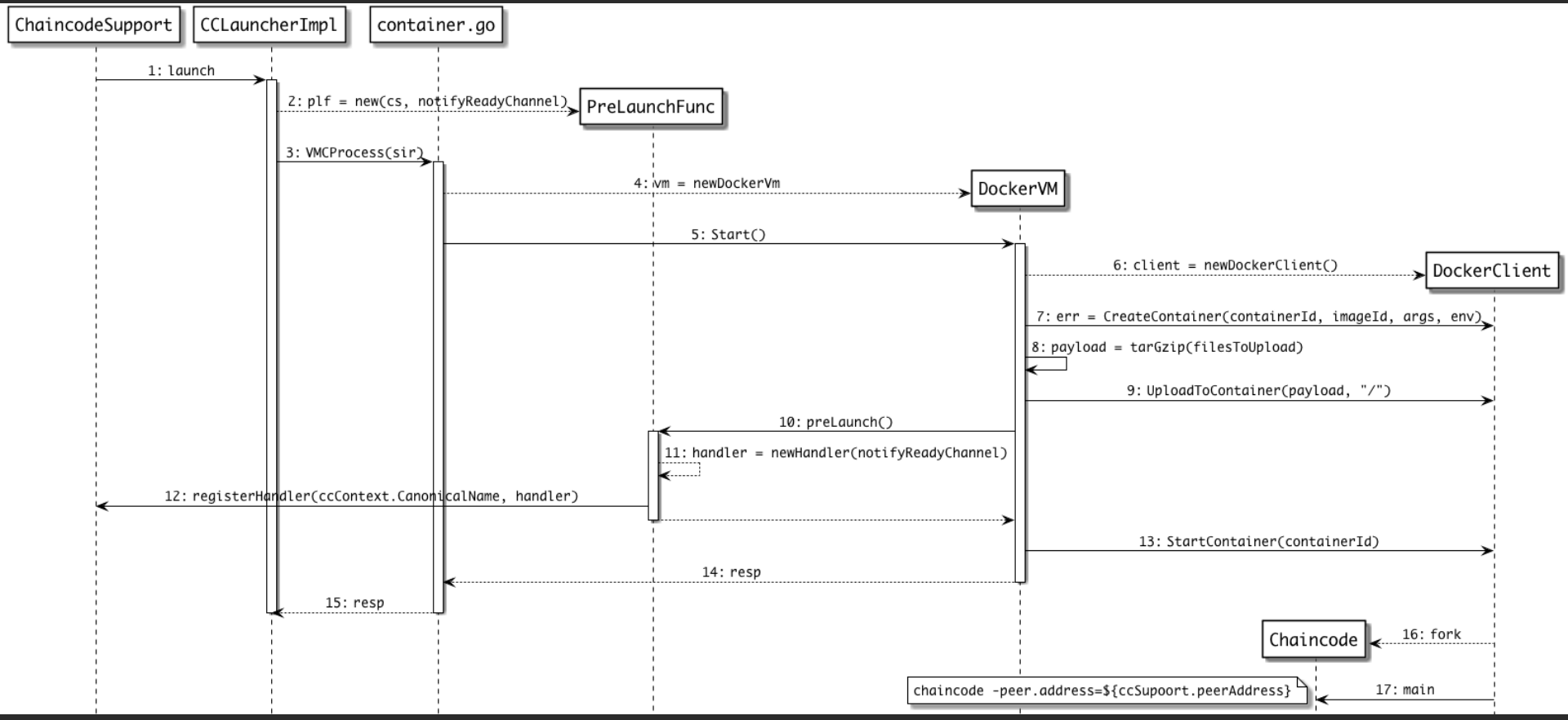
core.yaml: chaincode.golang.runtime: $(BASE_DOCKER_NS)/fabric-baseos:$(ARCH)-$(BASE_VERSION)

Chaincode Instantiate (Launch)

- Sequence Diagrams
  1. Prepare for StartImageReq
  2. Generate chaincode binary
  3. Start chaincode container (peer)
  4. Start chaincode process (inside container)

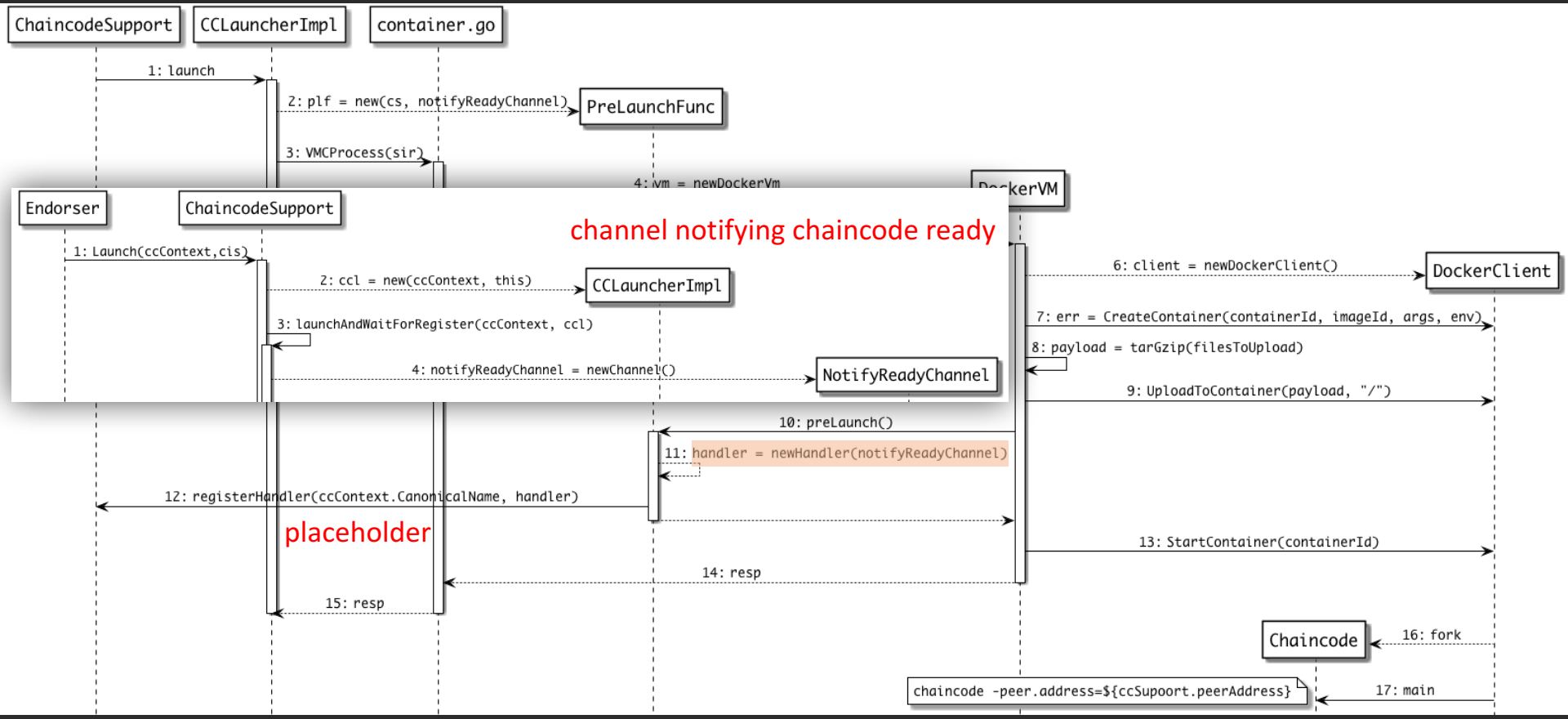
Chaincode Instantiate (Launch)

- Sequence Diagram: Start chaincode container (peer)



Chaincode Instantiate (Launch)

- Sequence Diagram: Start chaincode container (peer)



Chaincode Instantiate (Launch)

- Sequence Diagram: Start chaincode container (peer)
  - ChaincodeSupport.registerHandler()

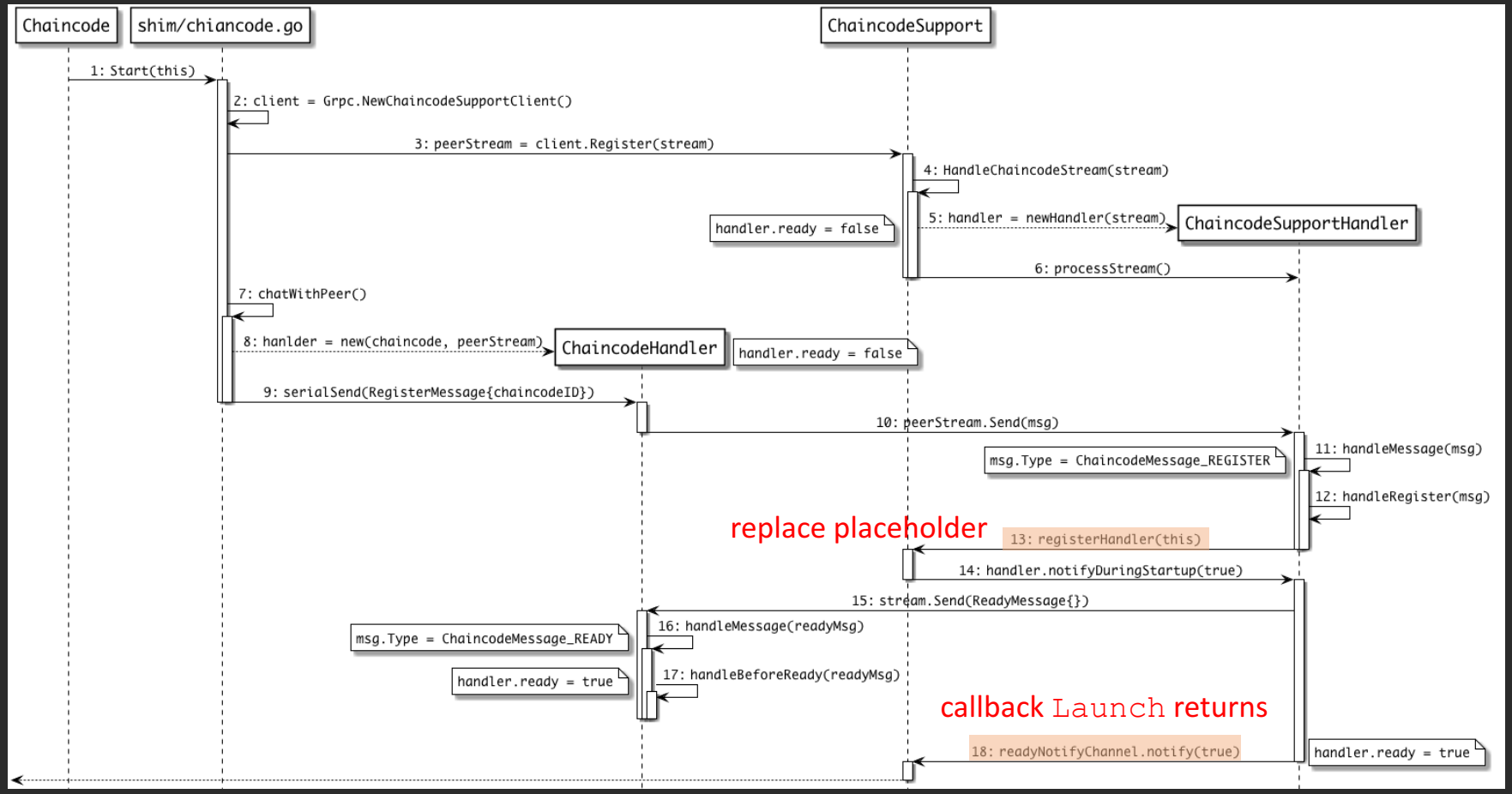
```
func (chaincodeSupport *ChaincodeSupport) registerHandler(chaincodehandler *Handler) error {
    key := chaincodehandler.ChaincodeID.Name

    chaincodeSupport.runningChaincodes.Lock()
    defer chaincodeSupport.runningChaincodes.Unlock()

    chrte2, ok := chaincodeSupport.chaincodeHasBeenLaunched(key)
    if ok && chrte2.handler.registered == true {
        return newDuplicateChaincodeHandlerError(chaincodehandler)
    }
    if chrte2 != nil {  second time
        chaincodehandler.readyNotify = chrte2.handler.readyNotify
        chrte2.handler = chaincodehandler
    } else {  first time
        if chaincodeSupport.userRunsCC == false {
            return errors.Errorf( format: "peer will not accept external chaincode connection %v (except in dev
 mode)", chaincodehandler.ChaincodeID)
        }
        chaincodeSupport.runningChaincodes.chaincodeMap[key] = &chaincodeRTEnv{handler: chaincodehandler}
    }
    chaincodehandler.registered = true
    chaincodehandler.txCtxs = make(map[string]*transactionContext)
    chaincodehandler.txidMap = make(map[string]bool)
    return nil
}
```

Chaincode Instantiate (Launch)

- Sequence Diagrams
  1. Prepare for StartImageReq
  2. Generate chaincode binary
  3. Start chaincode container (peer)
  4. Start chaincode process (inside container)

Chaincode Instantiate (Launch)

- Sequence Diagram: Start chaincode process



Chaincode Instantiate (Launch)

- Sequence Diagram: Start chaincode process
  - ChaincodeSupportHandler: core/chaincode/handler.go

```
type Handler struct {
    sync.Mutex
    //peer to shim grpc serializer. User only in serialSend
    serialLock  sync.Mutex
    ChatStream  ccintf.ChaincodeStream
    ChaincodeID *pb.ChaincodeID
    ccInstance  *sysccprovider.ChaincodeInstance

    chaincodeSupport *ChaincodeSupport
    ready            bool
    registered       bool
    readyNotify      chan bool

    //chan to pass error in sync and nonsync mode
    errc chan error
    // Map of txid to either invoke tx. Each tx will be
    // added prior to execute and remove when done execute
    txCtxs map[string]*transactionContext

    txidMap map[string]bool

    readyStateHandlers  map[pb.ChaincodeMessage_Type]func(*pb.ChaincodeMessage)
}

func (handler *Handler) getTxCtxId(chainID string, txid string) string {
    return chainID + txid
}
```

```
type transactionContext struct {
    chainID          string
    signedProp       *pb.SignedProposal
    proposal         *pb.Proposal
    responseNotifier chan *pb.ChaincodeMessage

    // tracks open iterators used for range queries
    queryIteratorMap map[string]commonledger.ResultsIterator

    txsimulator          ledger.TxSimulator
    historyQueryExecutor ledger.HistoryQueryExecutor
}
```

Chaincode Instantiate (Launch)

- Sequence Diagram: Start chaincode process
  - ChaincodeHandler: core/chaincode/shim/handler.go

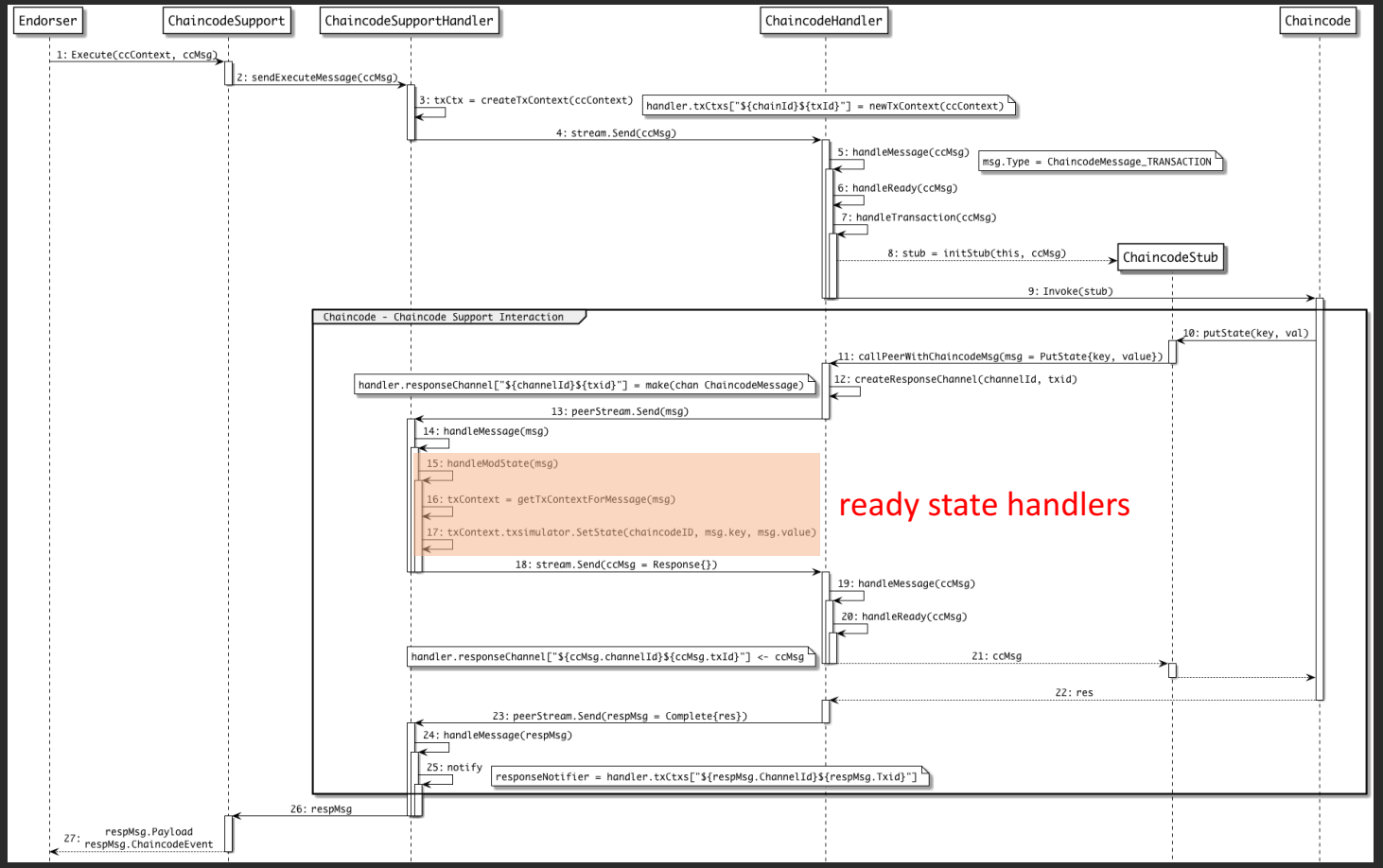
```
type Handler struct {
    //need lock to protect chaincode from attempting
    //concurrent requests to the peer
    sync.Mutex

    //shim to peer grpc serializer. User only in serialSend
    serialLock sync.Mutex

    ChatStream PeerChaincodeStream
    cc         Chaincode
    ready      bool
    // maps txid to the channel on which responses are communicated
    // by the shim to the chaincodeStub.
    responseChannel map[string]chan pb.ChaincodeMessage
}


func (handler *Handler) getTxCtxId(chainID string, txid string) string {
    return chainID + txid
}
```

Outline

- Components & Services
- Chaincode
  - Interfaces
  - Instantiation (Launch)
  - Query/Invocation
- Client SDK
- Fabric E2E Live Demo

Chaincode Invocation

- Sequence Diagram: transaction only putting state



Chaincode Invocation

- Sequence Diagram: transaction only putting state

```
v.readyStateHandlers = map[pb.ChaincodeMessage_Type]func(*pb.ChaincodeMessage){
    //events from CC at the end of a TX  that require notification
    pb.ChaincodeMessage_COMPLETED: v.notify,
    pb.ChaincodeMessage_ERROR:     v.notify,

    //state requests from CC that require processing
    pb.ChaincodeMessage_GET_STATE:           v.handleGetState,
    pb.ChaincodeMessage_GET_STATE_BY_RANGE:  v.handleGetStateByRange,
    pb.ChaincodeMessage_GET_QUERY_RESULT:    v.handleGetQueryResult,
    pb.ChaincodeMessage_GET_HISTORY_FOR_KEY: v.handleGetHistoryForKey,
    pb.ChaincodeMessage_QUERY_STATE_NEXT:    v.handleQueryStateNext,
    pb.ChaincodeMessage_QUERY_STATE_CLOSE:   v.handleQueryStateClose,
    pb.ChaincodeMessage_PUT_STATE:           v.handleModState,
    pb.ChaincodeMessage_DEL_STATE:           v.handleModState,
    pb.ChaincodeMessage_INVOKE_CHAINCODE:    v.handleModState,
}
```

```
type transactionContext struct {
    chainID          string
    signedProp       *pb.SignedProposal
    proposal         *pb.Proposal
    responseNotifier chan *pb.ChaincodeMessage

    // tracks open iterators used for range queries
    queryIteratorMap map[string]commonledger.ResultsIterator

    txsimulator          ledger.TxSimulator
    historyQueryExecutor ledger.HistoryQueryExecutor
}
```

Outline

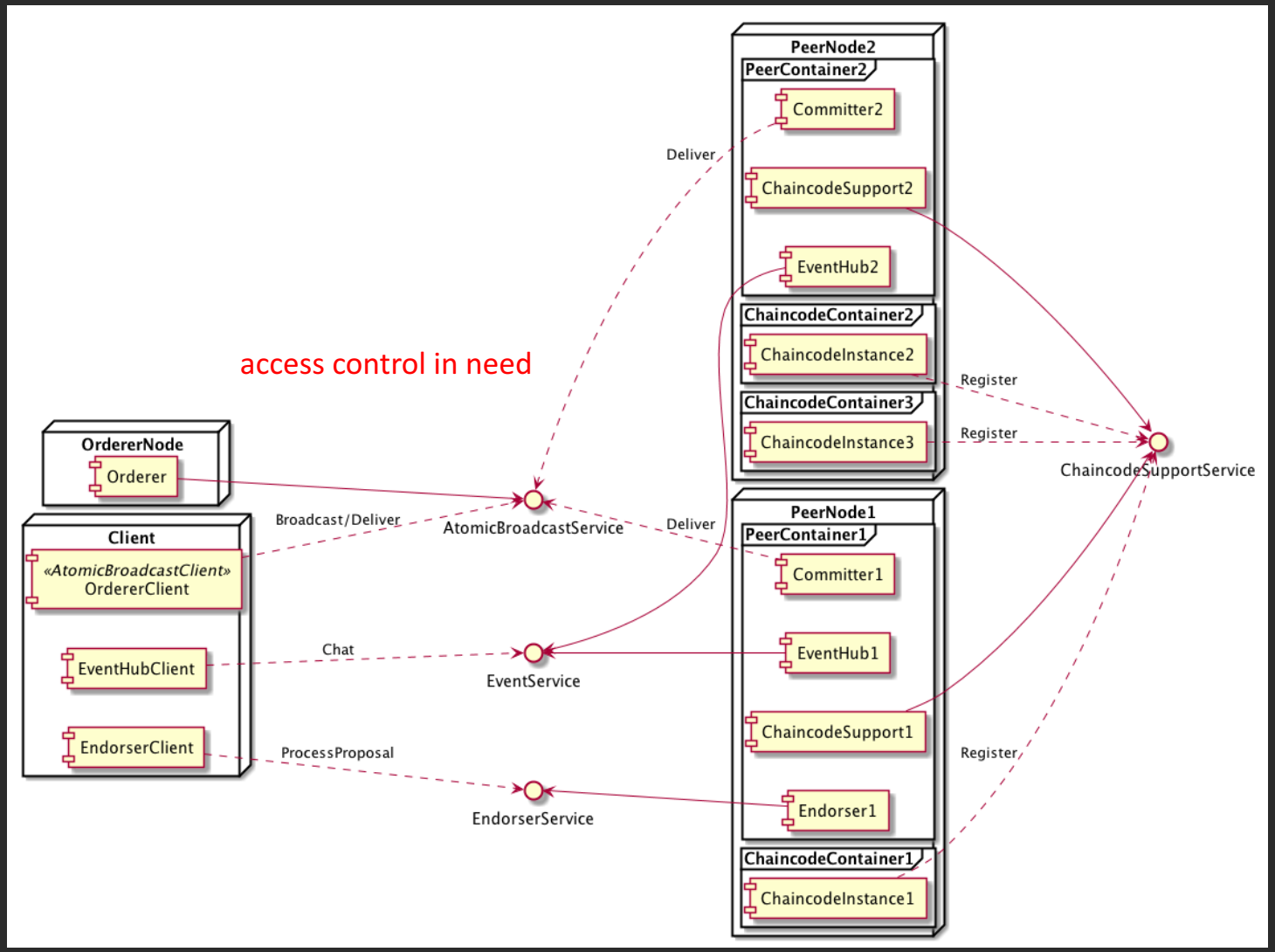
- Components & Services
- Chaincode
  - Interfaces
  - Instantiation (Launch)
  - Query/Invocation
- Client SDK
- Fabric E2E Live Demo

Client SDK

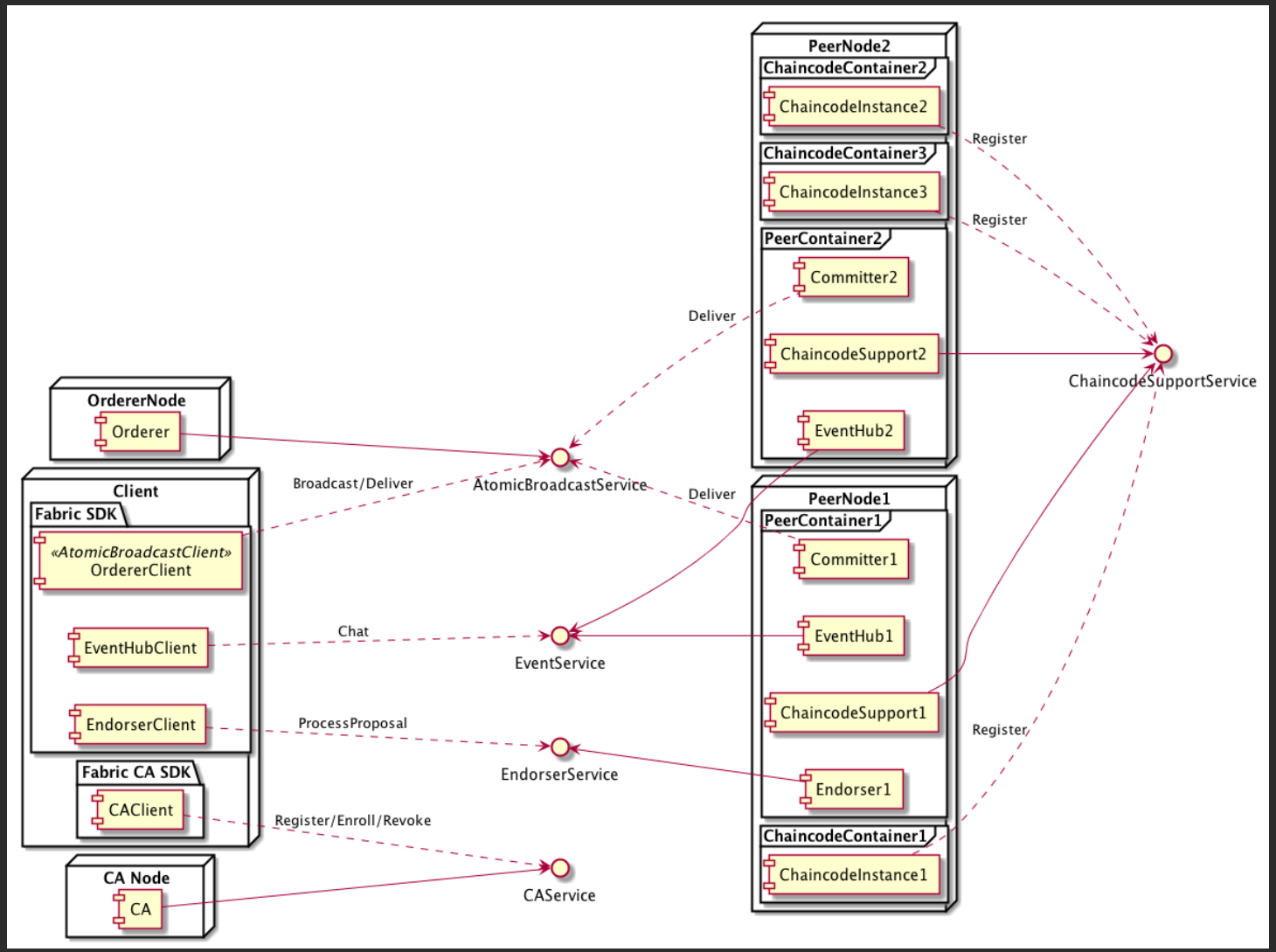
- Goal of Fabric SDK is to provide
  - the primitives to access the client-facing features in the blockchain network
  - a reasonable level of abstraction on top of that in the native language that the application is written in to ease the development effort

— From Fabric SDK Design Specification v1.0

Client SDK


PeerNode2
PeerContainer2
Committer2
Deliver
ChaincodeSupport2
EventHub2
ChaincodeContainer2
ChaincodeInstance2
Register
ChaincodeContainer3
ChaincodeInstance3
Register
ChaincodeSupportService
access control in need
OrdererNode
Orderer
Client
«AtomicBroadcastClient»
OrdererClient
Broadcast/Deliver
AtomicBroadcastService
Deliver
PeerNode1
PeerContainer1
Committer1
EventHubClient
Chat
EventService
EventHub1
ChaincodeSupport1
EndorserClient
ProcessProposal
EndorserService
Endorser1
Register
ChaincodeContainer1
ChaincodeInstance1

Client SDK


PeerNode2
ChaincodeContainer2
ChaincodeInstance2
ChaincodeContainer3
ChaincodeInstance3
PeerContainer2
Committer2
ChaincodeSupport2
EventHub2
Register
Register
Deliver
ChaincodeSupportService
OrdererNode
Orderer
Client
Fabric SDK
«AtomicBroadcastClient»
OrdererClient
EventHubClient
EndorserClient
Fabric CA SDK
CAClient
Broadcast/Deliver
AtomicBroadcastService
Deliver
PeerNode1
PeerContainer1
Committer1
EventHub1
ChaincodeSupport1
Endorser1
ChaincodeContainer1
ChaincodeInstance1
Chat
EventService
ProcessProposal
EndorserService
Register
Register/Enroll/Revoke
CA Node
CA
CAService

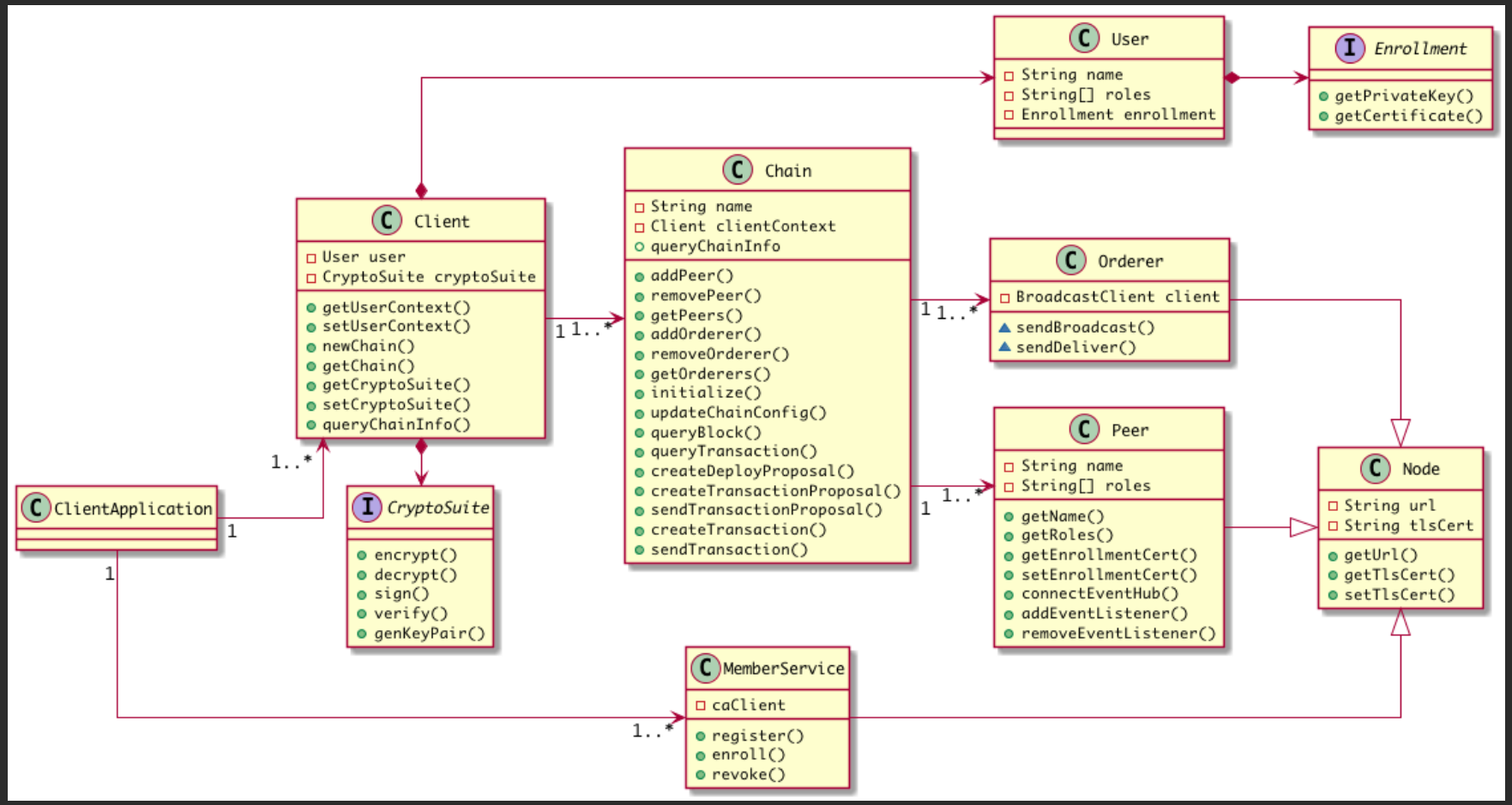
Client SDK

- Main Scenarios
  - Membership Registration and Enrollment
  - Chain Construction
  - Transaction/Query Support

- Key Modules
  - Lv0: Client, MemberService
  - Lv1: Chain
  - Lv2: Peer, Orderer, User
  - Lv3: Transaction, Proposal, Response, CryptoSuite

Client SDK

- Class Diagram in Design Specification



Client SDK

- Java Implementation Mapping

Design Specification	Java Implementation
Client	org.hyperledger.fabric.sdk.HFClient
CryptoSuite	org.hyperledger.fabric.sdk.security.CryptoSuite
Chain	org.hyperledger.fabric.sdk.Channel
MemberService	org.hyperledger.fabric_ca.sdk.HFCAClient
User	org.hyperledger.fabric.sdk.User
Orderer	org.hyperledger.fabric.sdk.Orderer
Peer	org.hyperledger.fabric.sdk.Peer
	org.hyperledger.fabric.sdk.EventHub
Node	—
Transaction	org.hyperledger.fabric.sdk.TransactionRequest
Response	org.hyperledger.fabric.sdk.ProposalResponse

Client SDK - Main Scenarios

- Membership Registration and Enrollment
- Chain Construction
- Transaction/Query Support

Client SDK

- Scenario - Membership Registration and Enrollment
  - HFCAClient.register()

```
public String register(RegistrationRequest request, User registrar)
        throws RegistrationException, InvalidArgumentException {
    /* omit check */
    setUpSSL();
    try {
        String body = request.toJson();
        String authHdr = getHTTPAuthCertificate(registrar.getEnrollment(), body);
        JsonObject resp = httpPost( url: url + HFCA_REGISTER, body, authHdr);
        String secret = resp.getString( s: "secret");
        if (secret == null) {
            throw new Exception("secret was not found in response");
        }
        return secret;
    } catch (Exception e) {
        throw new RegistrationException(format("Error while registering the user %s url: %s  %s ",
                registrar, url, e.getMessage()), e);
    }
}
```

Client SDK

- Scenario - Membership Registration and Enrollment
  - HFCAClient.enroll()

```
public Enrollment enroll(String user, String secret, EnrollmentRequest req)
        throws EnrollmentException, InvalidArgumentException {
    /* omit check */
    setUpSSL();
    try {
        if (req.getKeyPair() == null) {  req.setKeyPair(cryptoSuite.keyGen()); }
        if (req.getCsr() == null) { req.setCSR(cryptoSuite.generateCertificationRequest(user, req.getKeyPair())); }
        if (caName != null && !caName.isEmpty()) { req.setCAName(caName); }
        String body = req.toJson();
        String responseBody = httpPost( url: url + HFCA_ENROLL, body, new UsernamePasswordCredentials(user, secret));
        JsonObject jsonst = (JsonObject) Json.createReader(new StringReader(responseBody)).read();
        JsonObject result = jsonst.getJsonObject("result");
        if (result == null) {
            throw new EnrollmentException(format("FabricCA failed enrollment for user %s - " +
                    "response did not contain a result", user));
        }
        String signedPem = new String(Base64.getDecoder().decode(result.getString( s: "Cert").getBytes(UTF_8)));
        return new HFCAEnrollment(req.getKeyPair(), signedPem);
    } catch (EnrollmentException ee) {
        throw ee;
    } catch (Exception e) {
        throw new EnrollmentException(format("Url:%s, Failed to enroll user %s ", url, user), e);
    }
}
```

Client SDK - Main Scenarios

- Membership Registration and Enrollment
- Chain Construction
- Transaction/Query Support

Client SDK

- Scenario - Chain/Channel Construction
  - Channel.createNewInstance()

```
static Channel createNewInstance(String name, HFClient clientContext) throws InvalidArgumentException {
    return new Channel(name, clientContext);
}

static Channel createNewInstance(String name, HFClient hfClient, Orderer orderer,
                                 ChannelConfiguration channelConfiguration, byte[]... signers)
        throws InvalidArgumentException, TransactionException {
    return new Channel(name, hfClient, orderer, channelConfiguration, signers);
}


private Channel(String name, HFClient client, final boolean systemChannel) throws InvalidArgumentException {
    /* omit check */
    this.systemChannel = systemChannel;
    if (systemChannel) {
        name = SYSTEM_CHANNEL_NAME; //It's special !
        initialized = true;
    }
    this.name = name;
    this.client = client;
    logger.debug(format("Creating channel: %s, client context %s", isSystemChannel() ? "SYSTEM_CHANNEL" : name,
            client.getUserContext().getName()));
}
```

Client SDK

- Scenario - Chain/Channel Construction
  - Channel.createNewInstance()

```
private Channel(String name, HFClient hfClient, Orderer orderer, ChannelConfiguration channelConfiguration,
                byte[][] signers) throws InvalidArgumentException, TransactionException {
    this(name, hfClient, systemChannel: false);
    /* omit check */
    addOrderer(orderer);
    Envelope ccEnvelope = Envelope.parseFrom(channelConfiguration.getChannelConfigurationAsBytes());
    final Payload ccPayload = Payload.parseFrom(ccEnvelope.getPayload());
    final ChannelHeader ccChannelHeader = ChannelHeader.parseFrom(ccPayload.getHeader().getChannelHeader());
    if (ccChannelHeader.getType() != HeaderType.CONFIG_UPDATE.getNumber()) {
        throw new InvalidArgumentException(format("Creating channel; %s expected config block type %s, " +
                        "but got: %s", name, HeaderType.CONFIG_UPDATE,
                        HeaderType.forNumber(ccChannelHeader.getType())));
    }
    final ConfigUpdateEnvelope configUpdateEnv = ConfigUpdateEnvelope.parseFrom(ccPayload.getData());
    ByteString configUpdate = configUpdateEnv.getConfigUpdate();

    sendUpdateChannel(configUpdate.toByteArray(), signers, orderer);
    genesisBlock = getGenesisBlock(orderer); // get Genesis block to make sure channel was created.
    logger.debug(format("Created new channel %s on the Fabric done.", name));
}
```

Client SDK

- Scenario - Chain/Channel Construction
  - Channel.joinPeer()

```
public Channel joinPeer(Peer peer, PeerOptions peerOptions) throws ProposalException {
    logger.debug(format("Channel %s joining peer %s, url: %s", name, peer.getName(), peer.getUrl()));
    /* omit check */
    //channel is not really created and this is targeted to system channel
    final Channel systemChannel = newSystemChannel(client);
    TransactionContext transactionContext = systemChannel.getTransactionContext();
    FabricProposal.Proposal joinProposal = JoinPeerProposalBuilder.newBuilder()
            .context(transactionContext).genesisBlock(genesisBlock).build();
    SignedProposal signedProposal = getSignedProposal(transactionContext, joinProposal);
    addPeer(peer, peerOptions); //need to add peer.
    Collection<ProposalResponse> resp = sendProposalToPeers(Collections.singletonList(peer),
            signedProposal, transactionContext);
    ProposalResponse pro = resp.iterator().next();
    if (pro.getStatus() == ProposalResponse.Status.SUCCESS) {
        logger.info(format("Peer %s joined into channel %s", peer.getName(), name));
    } else {
        removePeer(peer);
        throw new ProposalException(format("Join peer to channel %s failed.  Status %s, details: %s",
                name, pro.getStatus().toString(), pro.getMessage()));
    }
}
```

CSCCProposal

Client SDK

- Scenario - Chain/Channel Construction
  - Channel.initialize()

```
public Channel initialize() throws InvalidArgumentException, TransactionException {
    /* omit check */   MSP Config
    parseConfigBlock(); // Parse config block for this channel to get it's information.
    loadCACertificates();  // put all MSP certs into cryptoSuite
    startEventQue(); //Run the event for event messages from event hubs.
    logger.debug(format("Eventque started %s", "" + eventQueueThread));

    for (EventHub eh : eventHubs) { //Connect all event hubs
        eh.connect(getTransactionContext());
    }
    logger.debug(format("%d eventhubs initialized", getEventHubs().size()));

    registerTransactionListenerProcessor(); //Manage transactions.
    logger.debug(format("Channel %s registerTransactionListenerProcessor completed", name));

    this.initialized = true;
    logger.debug(format("Channel %s initialized", name));
    return this;
}
```

Client SDK - Main Scenarios

- Membership Registration and Enrollment
- Chain Construction
- Transaction/Query Support

Client SDK

- Scenario - Query/Submit Transaction
  - Channel.sendProposal()

```
public Collection<ProposalResponse> sendTransactionProposal(TransactionProposalRequest transactionProposalRequest,
                                                            Collection<Peer> peers)
        throws ProposalException, InvalidArgumentException {
    return sendProposal(transactionProposalRequest, peers);
}
public Collection<ProposalResponse> queryByChaincode(QueryByChaincodeRequest queryByChaincodeRequest,
                                                     Collection<Peer> peers)
        throws InvalidArgumentException, ProposalException {
    return sendProposal(queryByChaincodeRequest, peers);
}
private Collection<ProposalResponse> sendProposal(TransactionRequest proposalRequest, Collection<Peer> peers)
        throws InvalidArgumentException, ProposalException {
    /* omit check */
    proposalRequest.setSubmitted();
    TransactionContext transactionContext = getTransactionContext(proposalRequest.getUserContext());
    transactionContext.verify(proposalRequest.doVerify());
    transactionContext.setProposalWaitTime(proposalRequest.getProposalWaitTime());
    // Protobuf message builder
    ProposalBuilder proposalBuilder = ProposalBuilder.newBuilder();
    proposalBuilder.context(transactionContext);
    proposalBuilder.request(proposalRequest);
    SignedProposal invokeProposal = getSignedProposal(transactionContext, proposalBuilder.build());
    return sendProposalToPeers(peers, invokeProposal, transactionContext);
}
```

Client SDK

- Scenario - Query/Submit Transaction
  - Channel.sendTransaction()

```
public CompletableFuture<TransactionEvent> sendTransaction(Collection<ProposalResponse> proposalResponses,
                                                           Collection<Orderer> orderers, User userContext) {
    /* omit check */
    Envelope transactionEnvelope = createTransactionEnvelope(proposalResponses, userContext);
    CompletableFuture<TransactionEvent> sret = registerTxListener(proposalTransactionID);
    boolean success = false; Exception lException = null; BroadcastResponse resp = null;
    for (Orderer orderer : orderers) {
        try {
            resp = orderer.sendTransaction(transactionEnvelope);
            lException = null; // no longer last exception .. maybe just failed.
            if (resp.getStatus() == Status.SUCCESS) { success = true; break; }
        } catch (Exception e) {
            lException = new Exception(format("Channel %s unsuccessful sendTransaction to orderer %s (%s). %s",
                    name, orderer.getName(), orderer.getUrl(), getRespData(resp)), e);
        }
    }
    if (success) { return sret; } else {
        String emsg = format("Channel %s failed to place transaction %s on Orderer. Cause: UNSUCCESSFUL. %s",
                name, proposalTransactionID, getRespData(resp));
        unregisterTxListener(proposalTransactionID);
        CompletableFuture<TransactionEvent> ret = new CompletableFuture<>();
        ret.completeExceptionally(lException != null ? new Exception(emsg, lException) : new Exception(emsg));
        return ret;
    }
}
```

Conclusion

- Components & Services
  - gRPC Service Interfaces between Fabric Components
- Chaincode Interfaces
- Chaincode Instaintiation & Invocation Logic
  - Sequence Diagram
  - Source Code
- Client SDK
  - Key Modules
  - Main Scenarios & Core Code

Outline

- Components & Services
- Chaincode
  - Interfaces
  - Instantiation (Launch)
  - Query/Invocation
- Client SDK
- Fabric E2E Live Demo

Fabric E2E Live Demo

- Fabric Network Settings
  - 1 peer & 1 orderer & 1 CA
- Sample Golang Chaincode
  - Using `vendor` to include third-party libraries
- Sample Java Client SDK Application
  - Joining already Existing Channel
  - Invoking & Querying Chaincode
- Source Code
  - https://github.com/dawnwords/fabric-demo

Thanks for Listening!

Q & A