

Desenvolvendo Dapps em plataforma Hyperledger Fabric

**CC-Tools Library
Hyperledger Labs**

Utilizando o readthedocs

<https://golegger-cc-tools.readthedocs.io/en/latest/>

GoLedger CC-Tools

Search docs

Home

Features

Getting Started

Tutorials

Assets

Datatypes

Events

Transactions

External Tools

Testing

Functions

Next »

Home

Welcome to CC-Tools documentation 🙌

The **GoLedger CC-Tools** library is a collaborative effort from GoLedger to provide developers a powerful and easy-to-use library for creating Hyperledger Fabric chaincodes.

Developed using GoLang, the **GoLedger CC-Tools** library has several features that facilitate the journey of learning, development and deployment in production of a chaincode.

GoLedger CC-Tools is an open-source project led by **GoLedger** and open for use to the Hyperledger development community.

Here's where you can start:

- [Getting Started](#)
- [Tutorials](#)
- Key concepts
 - [Assets](#)
 - [Datatypes](#)
 - [Events](#)



Verificando o docker

```
sudo systemctl enable docker
```

Limpando o docker

```
docker stop $(docker ps -a -q)
```

```
docker rm $(docker ps -a -q)
```

```
docker rmi -f $(docker images)
```

```
docker volume prune
```

```
docker system prune
```

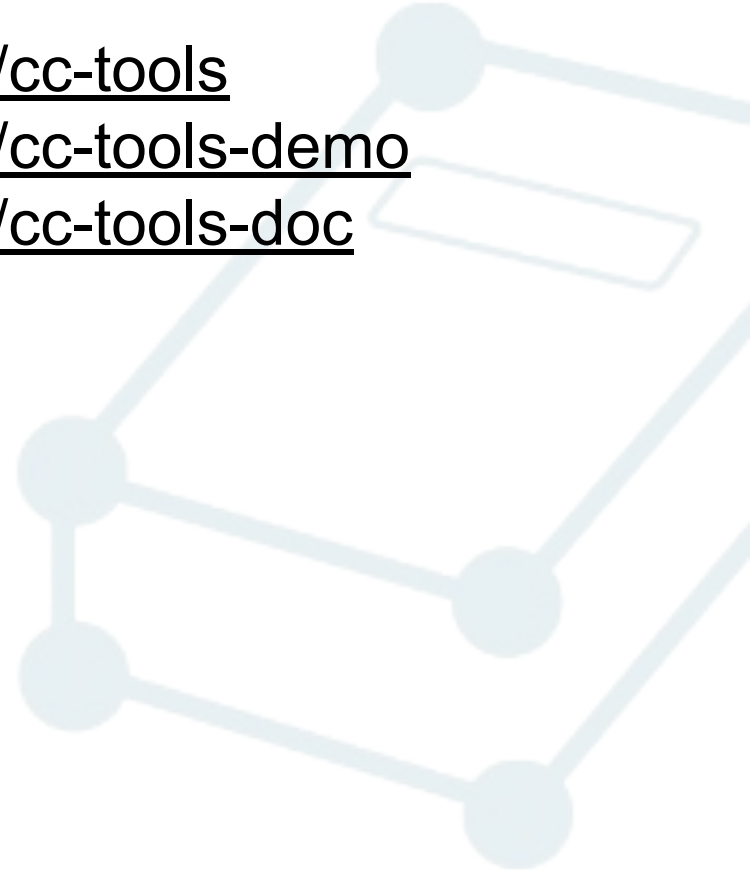


Repositórios oficiais

<https://github.com/hyperledger-labs/cc-tools>

<https://github.com/hyperledger-labs/cc-tools-demo>

<https://github.com/hyperledger-labs/cc-tools-doc>



Desenvolvendo chaincodes

Mapeamento de dados

Gerenciamento de direito de escrita e leitura entre organizações (MSP)

Cadastramento de transações



Biblioteca CC-Tools

A biblioteca CC-Tools é desenvolvida em GoLang e possui diversas características que facilitam a jornada de aprendizado, desenvolvimento e deployment em produção de um chaincode para Hyperledger Fabric.

Biblioteca CC-Tools - características

Open source

Padronização de assets, keys e referências de assets (asset dentro de asset)

Tipos de dados padrão e customizáveis

Gerenciamento de organizações (MSP)

Cadastramento de Transações

Gerenciamento de permissões de escrita por propriedade de assets por MSP

Gerenciamento de private data.

Biblioteca CC-Tools - características

Transações embedded:

- Create
- Read
- Update
- Delete
- Search (paginated)
- ReadHistory

Transações customizáveis

Integração das transações com a Rest API (GET, PUT, POST, DELETE)

Permissionamento de chamadas de transações por MSP

Exemplo de Asset

```
// Description of a book
var Book = assets.AssetType{
  Tag: "book",
  Label: "Book",
  Description: "Book",
  Props: []assets.AssetProp{
    {
      IsKey: true, // Primary Key
      Tag: "title",
      Label: "Book Title",
      DataType: "string",
      Writers: []string{"org2msp"}, // This means only org2
      can create the asset (others can edit)
    },
    {
      Tag: "currentTenant",
      Label: "Current Tenant",
      DataType: "->person", /// Reference to another asset
    },
    {
      Tag: "genres",
      Label: "Genres",
      DataType: "[]string", // String list
    },
    {
      Tag: "published",
      Label: "Publishment Date",
      DataType: "datetime", // Date property
    },
  },
}
```

Exemplo de Transação

```
// Updates the tenant of a Book
// POST Method
var UpdateBookTenant = tx.Transaction{
    Tag: "updateBookTenant",
    Label: "Update Book Tenant",
    Description: "Change the tenant of a book",
    Method: "PUT",
    Callers: []string{"$org\DNSP"}, // Any orgs can call
    this transaction

    Args: []tx.Argument{
        {
            Tag: "book",
            Label: "Book",
            Description: "Book",
            DataType: "->book",
            Required: true,
        },
        {
            Tag: "tenant",
            Label: "tenant",
            Description: "New tenant of the book",
            DataType: "->person",
        },
    },
    Routine: func(stub *sw.StubWrapper, req
map[string]interface{}) ([]byte, errors.ICCError) {
        .....
        .....
        .....
        return bookJSON, nil
    },
}
```

Ferramentas

Testes unitários

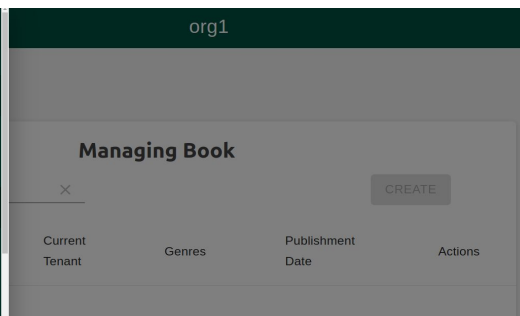
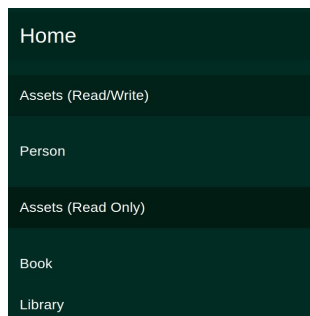
```
func TestCreateNewLibrary(t *testing.T) {
    stub := mock.NewMockStub("org3MSP",
        new(cc.CCDemo))

    expectedResponse :=
        map[string]interface{}{
            "@key":
                "library:3cab201f-9e2b-579d-b7b2-72297ed1
                7f49",
            "@lastTouchBy": "org3MSP",
            "@lastTx": "createNewLibrary",
        }
}
```

Web service (Rest API
integrado



Aplicação Web



Repositório cc-tools-demo

Baixar o repositório

git clone <https://github.com/hyperledger-labs/cc-tools-demo>

Vendorar o chaincode no diretório *chaincode*

go mod vendor

Iniciando o ambiente

Executar o script para HL Fabric 2.2 com 3 orgs

./startDev.sh

Executar o script para HL Fabric 1.4 com 1 org

./startDev.sh -n 1

Para subir a aplicação Web

./run-web-cc.sh

cc-tools-demo – diretórios e scripts

```
-- chaincode      # chaincode source code
  |-- assettypes  # definições dos assets
  |-- datatypes   # tipos de dados customizados
  |-- txdefs      # transações customizadas
-- rest-server     # web service (REST API)
  npmInstall.sh   # vendorar web service
  startDev.sh      # reiniciar apis (1.4)
  startDev2.sh     # reiniciar apis (2.2)
-- fabric          # artefatos HL Fabric 1.4
  startDev.sh      # iniciar rede HL 1.4
-- fabric2         # artefatos HL Fabric 2.2
  startDev.sh      # iniciar rede HL 2.2
renameProject.sh  # renomear nome do chaincode
startDev.sh       # iniciar rede HL Fabric 1.4
startDev2.sh      # iniciar rede HL Fabric 2.2
```

Definição de um asset

Definição dos assets dentro da pasta *assettypes*

```
var Person = assets.AssetType{
  Tag:      "person", // Identificação do ativo (JSON)
  Label:    "Person", // Texto identificador do ativo
  Description: "Personal data", // Texto identificador detalhado do ativo
  Props: []assets.AssetProp{
    {
      IsKey: true, // Propriedade faz parte da chave primária/composta
      Tag:   "id", // Identificação da propriedade (JSON/interface{})
      Label: "CPF (Brazilian ID)", // Texto identificador da propriedade
      DataType: "cpf", // Tipo da propriedade (embedded ou custom)
      Writers: []string{ "org1MSP" }, // Organização que pode criar/alterar a propriedade
    },
    {
      // Mandatory property
      Required: true,
      Tag:      "name",
      Label:    "Name of the person",
      DataType: "string",
      Validate: func(name interface{}) error { // Função de validação (criação ou alteração da propriedade)
        nameStr := name.(string)
        if nameStr == "" {
          return fmt.Errorf("name must be non-empty")
        }
        return nil
      },
    },
  },
}
```

Lista de assets do chaincode

Os assets devem estar explicitamente definidos no arquivo *assetTypeList.go* na pasta *chaincode*

```
var assetTypeList = []assets.AssetType{
    assettypes.Person,
    assettypes.Book,
    assettypes.Library,
    assettypes.Secret,
}
```


Datatypes embedded

CC-Tools possui os seguintes *datatypes* padrão

string # texto livre

number # numero flutuante

datetime # data e hora

boolean # true / false

[]string # array de texto livre

[]number # array numero flutuante

[]datetime # array data e hora

[]boolean # array de true / false

->[[asset](#)] # referencia a outro asset

[]->[[asset](#)] # array de referencias a assets

Definição de um Datatype custom

Definição dos *custom datatypes* dentro da pasta *datatypes*

```
var cpf = assets.DataType{
  Parse: func(data interface{}) (string, interface{}, errors.ICCError) {
    cpf, ok := data.(string)
    if !ok {
      return "", nil, errors.NewCCError("property must be a string", 400)
    }

    cpf = strings.ReplaceAll(cpf, ".", "")
    cpf = strings.ReplaceAll(cpf, "-", "")
    if len(cpf) != 11 {
      return "", nil, errors.NewCCError("CPF must have 11 digits", 400)
    }

    ...
    return cpf, cpf, nil
  },
}
```

Lista de custom datatypes do chaincode

Os custom datatypes devem estar explicitamente definidos no arquivo ***datatypes.go*** na pasta *chaincode/datatypes*

```
var CustomDataTypes = map[string]assets.DataType{  
    "cpf":    cpf,  
    "bookType": bookType,  
}
```

Transações embedded

CC-Tools possui os seguintes transações embutidas automaticamente para uso.

```
Tx.ReadAsset           // Ler ativo no world state  
tx.CreateAsset // Criar no ativo no channel  
tx.UpdateAsset        // Atualizar ativo no channel  
tx.DeleteAsset // Deletar ativo no channel  
tx.Search             // Procurar ativos com rich query no world state  
tx.ReadAssetHistory   // Histórico de um ativo no ledger
```

Definição de uma transação custom

```
var CreateNewLibrary = tx.Transaction{
```

```
    Tag:      "createNewLibrary",           // Identificação da transação (API endpoint)
```

```
    Label:   "Create New Library",         // Texto identificador da transação
```

```
    Description: "Create a New Library", // Texto identificador detalhado da transação
```

```
    Method:  "POST",                      // Método do web service
```

```
    Callers: []string{"$orgMSP", "$orgMSP"}, // Organizações (MSP) que podem chamar essa transação
```

```
    Args: []tx.Argument{
```

```
    {
```

```
        Tag:      "name",                  // Identificação do argumento
```

```
        Label:   "Name",                  // Texto identificador do argumento
```

```
        Description: "Name of the library", // Texto identificador detalhado do argumento
```

```
        DataType: "string",              // Tipo de dados do argumento
```

```
        Required: true,                   // Argumento obrigatório (default=false)
```

```
    },
```

```
},
```

```
Routine: func(stub *sw.StubWrapper, req map[string]interface{}) ([]byte, errors.ICCError) {
```

```
    name, _ := req["name"].(string)           // Argumento do chamada
```

```
    libraryMap := make(map[string]interface{})
```

```
    libraryMap["@assetType"] = "library"
```

```
    libraryMap["name"] = name
```

```
    libraryAsset, err := assets.NewAsset(libraryMap) // Preparação do ativo para gravação
```

```
    if err != nil {...}
```

```
    _err = libraryAsset.PutNew(stub)           // Criação do transaction proposal response
```

```
    if err != nil {...}
```

```
    libraryJSON, nerr := json.Marshal(libraryAsset)
```

```
    if nerr != nil {...}
```

```
    return libraryJSON, nil                   // Retorna resultado para a API
```

```
},
```

Lista de transações do chaincode

As transações, inclusive as embedded devem estar explicitamente definidas no arquivo ***txList.go*** na pasta ***chaincode***

```
var txList = []tx.Transaction{
    tx.CreateAsset,
    tx.UpdateAsset,
    tx.DeleteAsset,
    txdefs.CreateNewLibrary,
    txdefs.GetNumberOfBooksFromLibrary,
    txdefs.UpdateBookTenant,
    txdefs.GetBooksByAuthor,
}
```

Utilizando o Web Service padrão (API)

Web service padronizado para realizar chamadas na rede Blockchain.

Modelo padrão:

Invoke (realiza alteração no channel)

Métodos: *POST, PUT, DELETE*

api/invoke/:tx

Query (não altera o channel):

Métodos: *GET, POST*

api/query/:tx

Endpoints de apoio

query/getHeader

Método GET

Retorna versão do CC-Tools e dados do header para visualização na interface

query/getSchema

Métodos GET e POST

Retorna informações dos assets ou de um asset específico

query/getTx

Métodos GET e POST

Retorna informações das transações ou de uma transação específica

Exemplo de getSchema

POST /query/getSchema Obtem a descrição de um tipo de asset específico.

Parameters

Cancel

No parameters

Request body

application/json

O assetType deve conter um tipo de asset definido pelo chaincode.
Examples:
person

```
{  "assetType": "person"}
```

ExecuteClear

Responses

Request URL

http://localhost/api/query/getSchema

Server response

CodeDetails

200Response body

```
{ "tag": "person", "label": "Person", "description": "Personal data of someone", "props": [{"tag": "id", "label": "CPF (Brazilian ID)", "description": "", "isKey": true, "required": true, "readOnly": false, "dataType": "string", "writers": ["org1MSP", "org2MSP"]}, {"tag": "name", "label": "Name of the person", "description": "", "isKey": false, "required": true, "readOnly": false, "dataType": "string", "writers": null}, {"tag": "dateOfBirth", "label": "Date of Birth", "description": "", "isKey": false, "required": false, "readOnly": false, "dataType": "datetime", "writers": null}, {"tag": "height", "label": "Person's height", "description": "", "isKey": false, "required": false, "readOnly": false, "defaultValue": 0, "dataType": "number", "writers": null}]} 
```

Download

Response headers

Endpoints para as tx embedded

invoke/createAsset

Método POST

{"asset": [{JSON do ativo}, {JSON do ativo}, ...] }

invoke/readAsset

Método POST

{"key": {"@assetType": Tipo do ativo, Dados da chave } }

invoke/updateAsset

Método PUT

{"update": {"@assetType": Tipo do ativo, Dados da chave, Propriedades a serem alteradas } }

invoke/deleteAsset

Método DELETE

{"key": {"@assetType": Tipo do ativo, Dados da chave } }

Identificação de chaves dentro do asset

Propriedade da chave

@key

Asset deve ter a identificação do tipo do asset

@assetType

Para algumas operações a chave é calculada automaticamente.

Exemplo:

```
{ "key": { "@assetType": "person", "id": "318.207.920-48" } }
```

```
{ "key": { "@assetType": "person", "@key": "person:47061146-c642-51a1-844a-bf0b17cb5e19" } }
```

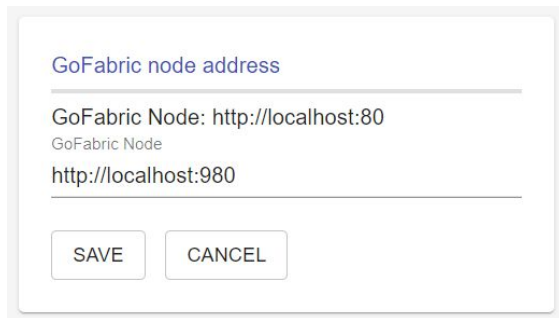
Utilizando a ferramenta Web App (Golnitus)

Dentro do diretório raiz

`./run-cc-web.sh`

Acessar o browser <http://localhost:8080>

Configurar a interface para acessar a api correta (icone de ferramenta do header)



A screenshot of a configuration dialog box titled "GoFabric node address". It contains two text input fields. The first field is labeled "GoFabric Node:" and contains the text "http://localhost:80". The second field is labeled "GoFabric Node" and contains the text "http://localhost:980". Below the input fields are two buttons: "SAVE" and "CANCEL".

GoFabric node address

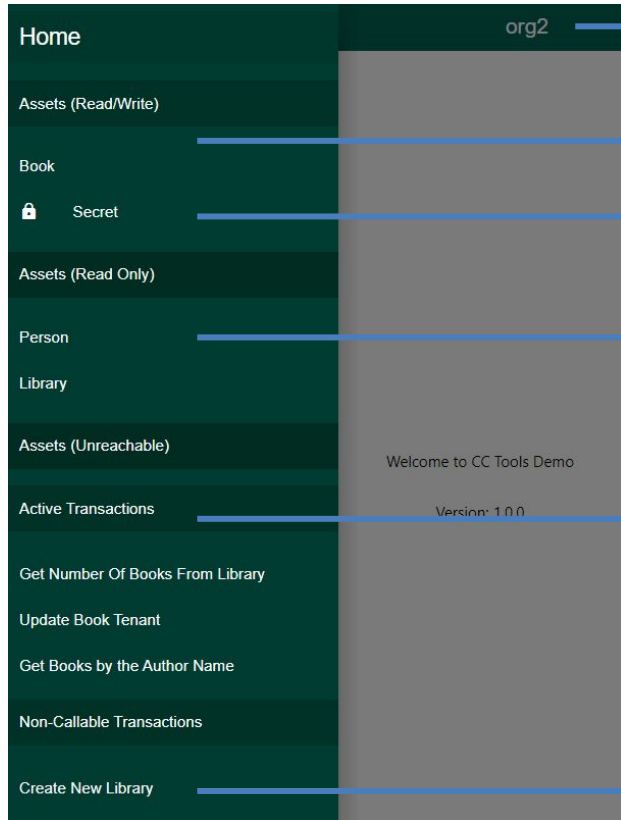
GoFabric Node: http://localhost:80

GoFabric Node

http://localhost:980

SAVE CANCEL

Usando o Golnitus



Organização (MSP) do web service

Ativos que podem ser criados pela organização

Ativos em private data (PVC)

Ativos que não podem ser criados (talvez possam ser atualizados ou deletados)

Transações que podem ser chamadas pela organização

Transações que não podem ser chamadas pela organização

Trabalhando com ativos

The screenshot shows a web application interface for 'Managing Person' under the 'org1' header. A 'CURL' button is highlighted with a blue arrow pointing to a separate box. A 'CREATE' button is also highlighted with a blue arrow pointing to a list of API operations. A blue arrow points from the 'Actions' column of the table to the same list of operations.

Managing Person

Search X

CREATE

CPF (Brazilian ID)	Name of the person	Date of Birth	Person's height	Actions
31820792048	Keyla	11/11/2000, 0:00:00 GMT-2	1.6	
12345678909	Maria		1.66	
65852796115	Carlos	11/11/1990, 0:00:00 GMT-2	1.8	

CURL

```
curl -X POST "http://localhost:80/api/query/search" -H 'Content-Type: application/json' -H 'cache-control: no-cache' -d '{"query":{"selector":["@assetType":"person"],"limit":11,"bookmark":"","resolve":true}}
```




CANCEL

Chamada para operações de

- CreateAsset*
- ReadAssetHistory*
- ReadAsset*
- UpdateAsset*
- DeleteAsset*

Documentação CC-Tools

<https://goledger-cc-tools.readthedocs.io/en/latest/>

 GoLedger CC-Tools

[Home](#)

[Welcome to CC-Tools documentation](#)

[Features](#)

[Getting Started](#)

[Tutorials](#)

[Assets](#)

[Transactions](#)

[External Tools](#)

[Testing](#)

[Functions](#)

```
# .readthedocs.yaml
build.tools:
  python: "3.11"
sphinx:
  configuration: conf.py
python.install:
  - requirements: pip.in
```

Supercharge your Sphinx docs with deployment on Read the Docs. **Sign up today!**

Ad by EthicalAds · Host ads

Next »

[Docs](#) » [Home](#)

Welcome to CC-Tools documentation 🙌

The **GoLedger CC-Tools** library is a collaborative effort from GoLedger to provide developers a powerful and easy-to-use library for creating Hyperledger Fabric chaincodes.

Developed using GoLang, the **GoLedger CC-Tools** library has several features that facilitate the journey of learning, development and deployment in production of a chaincode.

GoLedger CC-Tools is an open-source project lead by **GoLedger** and open for use to the Hyperledger development community.

Here's where you can start:

- [Getting Started](#)
- [Tutorials](#)
- Key concepts
 - [Assets](#)
 - [Transactions](#)
- [Reference guides](#)

This documentation page is in constant development...


Join our [Discord!](#)

Features

- Standard asset data mapping (and their properties)
- Encapsulation of Hyperledger Fabric chaincode sdk interface functions

Documentação Biblioteca GoLang

<https://pkg.go.dev/github.com/goledgerdev/cc-tools@v0.7.5>

 Search packages or symbols / 🔍

Why Go ▾ Learn Docs ▾ Packages C

Discover Packages > github.com/goledgerdev/cc-tools > assets 📄

assets package

Version: v0.7.5 Latest | Published: Sep 14, 2022 | License: Apache-2.0 | Imports: 16 | Imported by: 4

Details

✔ Valid go.mod file ? ✔ Redistributable license ? ✔ Tagged version ? ⊗ Stable version ? [Learn more](#)

Repository

github.com/goledgerdev/cc-tools

Links

🛡️ Report a Vulnerability 🔗 Open Source Insights

f

Documentation

Index

Constants

Variables

▶ Functions

▶ Types

Source Files

<> Documentation

Index

```
func CustomDataTypes(m map[string]DataType) error
func DataTypeMap() map[string]DataType
func GenerateKey(asset map[string]interface{}) (string, errors.ICCError)
func InitAssetList(l []AssetType)
func PutNewRecursive(stub *sw.StubWrapper, object map[string]interface{}) (map[string]interface{}, errors.ICCError)
func PutRecursive(stub *sw.StubWrapper, object map[string]interface{}) (map[string]interface{}, errors.ICCError)
func StartupCheck() errors.ICCError
func UpdateRecursive(stub *sw.StubWrapper, object map[string]interface{}) (map[string]interface{}, errors.ICCError)
```


Pacotes

O pacote *asset* é responsável pelas funções relativas ao gerenciamento de ativos no channel.

O pacote *transactions* é responsável pelas funções relativas as transações.

Pacotes

O pacote ***asset*** é responsável pelas funções relativas ao gerenciamento de ativos no channel.

Todas as funções do pacote ***asset*** validam as definições dos ativos criados no diretório *assettypes*.

Tipos (type) do pacote *asset*:

Asset : objeto *map[string]interface{}* de um ativo completo com as suas propriedades.

Key : objeto *map[string]interface{}* das informações da chave do asset.

O pacote ***transactions*** é responsável pelas funções relativas as transações.

O pacote *transaction* possui as transações embedded.

Criando/alterando ativos no channel

As principais funções para criar/alterar um *asset* (variação do *PutState*) são:

func NewAsset(m map[string]interface{}) (a Asset, err errors.ICCError)

Prepara o objeto tipo Asset para ser utilizado em operações futuras. Esperar

func (a *Asset) PutNew(stub *sw.StubWrapper) (map[string]interface{}, errors.ICCError)

Cria novo asset. Se asset já existir, retorna erro.

func (a *Asset) Put(stub *sw.StubWrapper) (map[string]interface{}, errors.ICCError)

Cria asset. Se asset já existir, atualiza

func (a *Asset) Update(stub *sw.StubWrapper, update map[string]interface{}) (map[string]interface{}, errors.ICCError)

Atualiza asset. Se asset não existir, retorna erro

Todas as transações de alteração de ativos validam as regras registradas na definição dos *assets* (*assettypes*)

Lendo ativos

As principais funções para ler um *asset* são (variações do *GetState*):

func NewAsset(m map[string]interface{}) (a Asset, err errors.ICCError)

Prepara o objeto tipo Asset para ser utilizando em operações futuras.

*func (k *Key) ExistsInLedger(stub *sw.StubWrapper) (bool, errors.ICCError)*

Verifica a existência de um asset.

*func (a *Asset) Get(stub *sw.StubWrapper) (*Asset, errors.ICCError)*

Busca um *asset*, caso exista, retorna em um objeto Asset

*func (k *Key) GetMap(stub *sw.StubWrapper) (map[string]interface{}, errors.ICCError)*

Busca um *asset*, caso exista, retorna em um objeto *map[string]interface{}*

*func (k *Key) GetBytes(stub *sw.StubWrapper) ([]byte, errors.ICCError)*

Busca um *asset*, caso exista, retorna em um objeto *[]byte*

*func (a *Asset) GetRecursive(stub *sw.StubWrapper) (map[string]interface{}, errors.ICCError)*

Busca um *asset*, caso exista, retorna em um objeto *map[string]interface{}* com todas as referências resolvidas.

Transação CC-Tools

Uma transação definida no CC-Tools possui as seguintes características.

Identificação para chamadas Rest API

Tag: "nome da transação"

Método da Rest API

Method: "POST" | "GET" | "PUT" | "DELETE"

Argumentos da transação

Args: []tx.Argument{
 {
 Tag: "nome do argumento",
 Label: "Descrição do argumento",
 Description: "Descrição detalhada do argumento",
 DataType: "tipo do argumento",
 Required: true,
 },...
}

Função para executar a transação

Routine: func(stub *sw.StubWrapper, req map[string]interface{}) ([]byte, errors.ICCError) {...

Lendo os argumentos recebidos na transação

Os argumentos enviados pela API podem ser lidos da seguinte forma:

variavel, _ := req["nome do argumento"].(tipo do argumento)

Exemplo:

*name, _ := req["name"].(string) // argumento name de tipo **string***

*name, ok := req["name"].(string) // argumento name verificando se está é **string***

*limit, _ := req["limit"].(float64) // argumento do tipo **number***

*birthDate, _ := req["birthdate"].(time.Time) // argumento do tipo **datetime***

*check, _ := req["check"].(bool) // argumento do tipo **boolean***

libraryKey, _ := req["library"].(assets.Key) // argumento do tipo referencia a um asset (->asset)

*names, _ := req["names"].([]string) // argumento name de tipo **[]string***

*limits, _ := req["limits"].([]float64) // argumento do tipo **[]number***

*birthDate, _ := req["birthdate"].([]time.Time) // argumento do tipo **[]datetime***

librarys, _ := req["librarys"].([]interface{}) // argumento do tipo array de referencias a um asset ([]->asset)

Retornando um resultado da transação

Caso a transação falhe, deve-se retornar um erro usando o pacote *errors*

Exemplos:

```
if !ok { errors.NewCCError("type parameter is missing or in wrong format", 400) }
```

```
if err { return nil, errors.WrapError(err, " parameter missing") }
```

Caso a função não tenha apresentado falhas, deve-se retornar um *[]byte*.

Exemplo:

```
ret, _ := json.Marshal(asset) // Retornando um json  
return ret, nil
```

Uma transação bem sucedida **NÃO** garante a atualização do channel.

Datatypes customizados

Datatypes podem ter tipos específicos para os chaincodes. São definidos na pasta datatypes

Um datatype custom possui as seguintes características:

AcceptedFormats: []string{"tipos de datatypes"} // opcional

DropDownValues: map[string]interface{}{lista do dropdown} //opcional

Description: // descrição do datatype custom

Parse: função de validação do datatype

Validando um valor com datatype custom

Dentro da função *Parse*, caso o valor tenha alguma inconsistência deve-se retornar erro utilizando o pacote *errors*

Exemplo:

```
return "", nil, errors.NewCCError("asset property must be an integer", 400)
```

Em caso de sucesso, deve retornar o valor em formato string e o valor real

Exemplo:

```
return fmt.Sprintf(retVal), retVal, err
```

Atualizando o chaincode

Para atualizar o chaincode em ambiente de validação deve usar o script:

```
upgradeCC.sh <version> <sequence>
```

Queries no channel

As buscas são feitas dentro do peer através de operações de pesquisa.

As pesquisas são realizadas nas bases de *state* escolhida (**LevelDb** ou **CouchDb**)

LevelDb é a base default e roda dentro do container peer

Pesquisas em **LevelDb** são realizadas para as chaves (primárias ou compostas)

CouchDb é um modelo mais poderoso e roda em um container separado do *peer*.

Rich queries e indexação são permitidas para **CouchDb**

Trabalhando com CouchDb

O uso do CouchDb pode ser realizado com a configuração dos endorsing peer

Váriáveis de ambiente:

`CORE_LEDGER_STATE_STATEDATABASE=CouchDB`

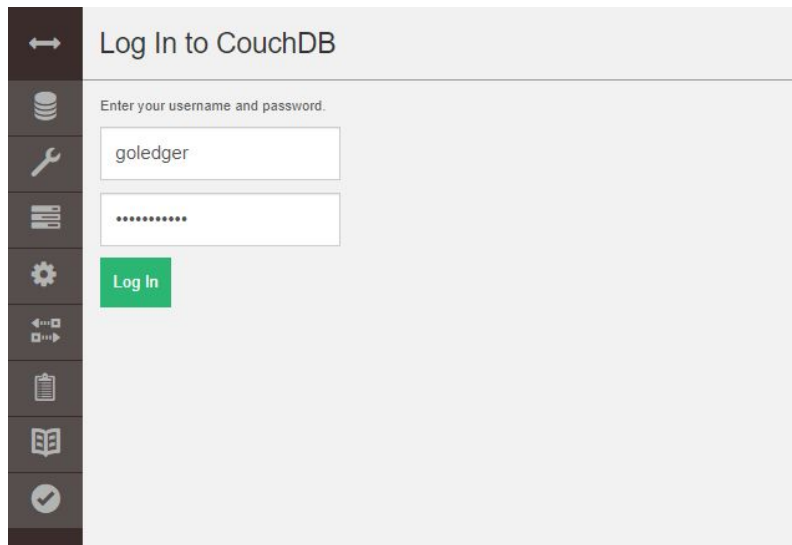
`CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHHDBADDRESS=couch.peer0.org.example.com:5984`

`CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME=admin`

`CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD=adminpw`

Exemplo de acesso ao couchdb via interface Web (Fauxion)

<http://localhost:5984/ utils/#login>



Log In to CouchDB

Enter your username and password.

Username: goledger

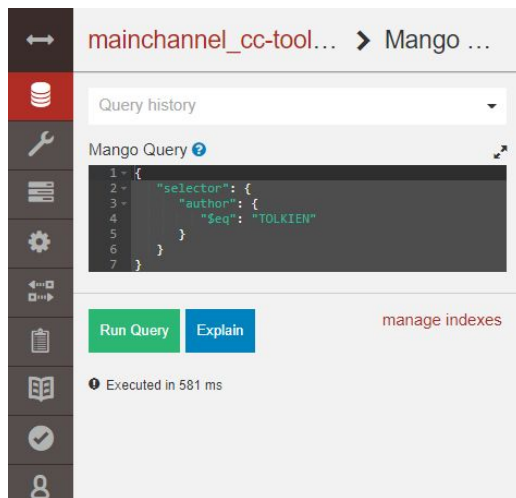
Password:

Log In

Usando a interface do CouchDB

O base do *state* é identificada pelo nome do *channel* junto com o nome do *chaincode*.

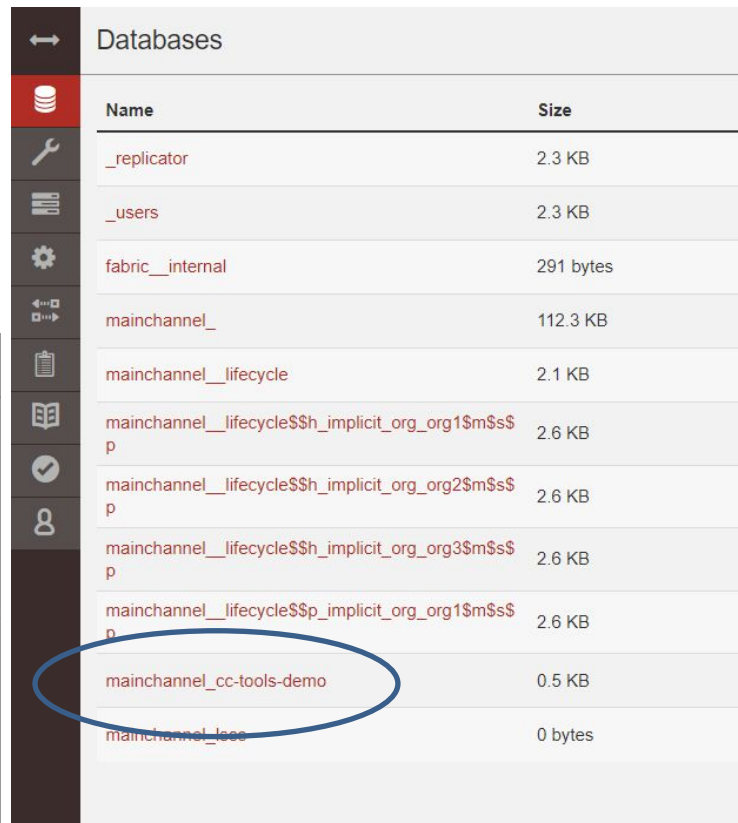
Queries podem ser executadas da interface através do item *Run Query with Mango*



The screenshot shows the Mango query interface. The top bar indicates the current database is 'mainchannel_cc-tool...' and the tool is 'Mango ...'. Below this, there's a 'Query history' dropdown. The main area is titled 'Mango Query' and contains a JSON query:

```
1 {
2   "selector": {
3     "author": {
4       "$eq": "TOLKIEN"
5     }
6   }
7 }
```

Below the query editor are two buttons: 'Run Query' (green) and 'Explain' (blue). To the right of these buttons is a link 'manage indexes'. At the bottom, it shows 'Executed in 581 ms'.



The screenshot shows the 'Databases' interface in CouchDB. It lists various databases and their sizes. The database 'mainchannel_cc-tools-demo' is circled in blue.

Name	Size
_replicator	2.3 KB
_users	2.3 KB
fabric__internal	291 bytes
mainchannel__	112.3 KB
mainchannel__lifecycle	2.1 KB
mainchannel__lifecycle\$\$h_implicit_org_org1\$m\$\$p	2.6 KB
mainchannel__lifecycle\$\$h_implicit_org_org2\$m\$\$p	2.6 KB
mainchannel__lifecycle\$\$h_implicit_org_org3\$m\$\$p	2.6 KB
mainchannel__lifecycle\$\$p_implicit_org_org1\$m\$\$p	2.6 KB
mainchannel_cc-tools-demo	0.5 KB
mainchannel__lifecycle	0 bytes

Tabela de Query CouchDb

Queries podem ser realizadas através de uma sintaxe. Maiores detalhes na documentação oficial do CouchDb

<https://docs.couchdb.org/en/3.2.2-docs/api/database/find.html>

Exemplo:

Busca documentos com status “draft”

```
"selector": {  
  "status": { "$eq": "draft" }  
}
```

Busca asset com year maior ou igual a 2020

```
"selector": {  
  "year": { "$gte": 1900 }  
}
```

Função Search

Função Search utilizando pesquisas realizadas no CouchDB.

Exemplo:

```
// Prepare couchdb query
```

```
query := map[string]interface{}{  
    "selector": map[string]interface{}{  
        "@assetType": "book",  
        "author":    authorName,  
    },  
}
```

```
response, err := assets.Search(stub, query, "", true)
```

```
...
```

Utilizando funções de pesquisa

O Hyperledger Fabric possui diversas funções para realizar queries dentro do chaincode. Essas funções trabalham com iterators dentro de loops.

Exemplos:

GetQueryResult

GetQueryResultWithPagination

GetStateByPartialCompositeKey

GetHistoryForKey

Exemplo:

```
resultsIterator, _ = stub.GetQueryResult(queryString)
```

```
for resultsIterator.HasNext() {
```

```
    queryResponse, err := resultsIterator.Next()
```

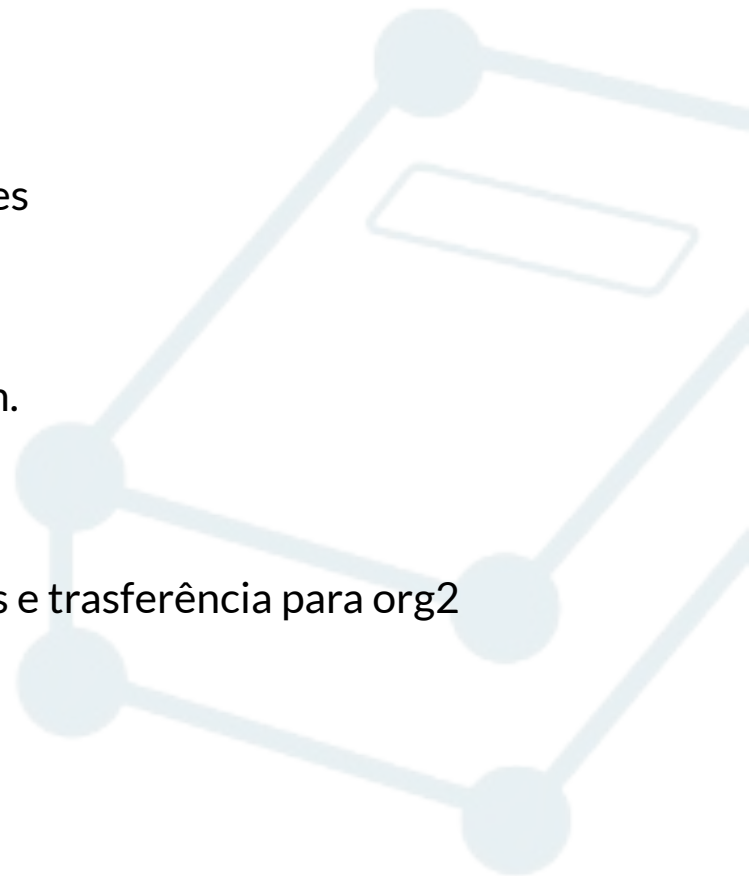
```
...
```


Tarefa



Tarefa Criando um Token

- Objetivo
- Começar com 1 org (`./startDev.sh -n 1`)
- Criar um token para transferencia entre organizações
 - Definitions do token
 - Método de criação de token
 - Método de transferência particionada do token.
 - Função de contabilidade do token.
 - Refazer a rede com 3 orgs
 - Limitar método de criação para org1 e métodos e trasferência para org2 (contabilidade sem restrições)



Definição do token

- Ativo Token
- Deve possuir os campos:
 - id (string) -> chave
 - proprietário (string) -> obrigatório
 - quantidade (number) -> default 0

Método criar token

- Método POST
- Argumentos -> iguais a definição do token
- Validação:
- Quantidade deve ser maior ou igual a 0



Método transferir token

- Método PUT
- Argumento destinatario (string)
- NÃO VERIFICAR quantidade
- Criar um novo token para o destinatário.
- Reduzir a quantidade do token origem

Método de contabilidade

- Método GET
- Argumento proprietatio
- Usar a função Search



Private Data Collections

A biblioteca CC-Tools possui mecanismos para trabalhar com PDCs.

Assets que terão as informações gravadas em PDCs devem ter o operador *Readers*.

Private Data Collection

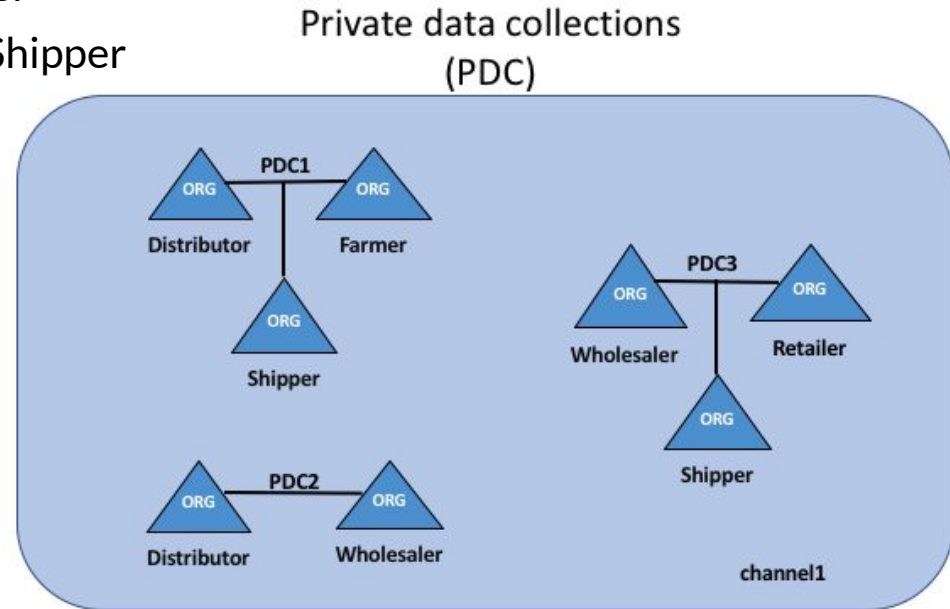
- Um *channel* pode conter informações acessíveis apenas a certas orgs.
- Essas informações ficam gravadas externamente ao *peer* dentro de bases do *CouchDb* em um banco de dados transiente.
- Dentro do *ledger* ficam registradas apenas os hashes das informações.
- Esse conceito é chamado de ***Private Data Collection***.
- As organizações que podem acessar a informação privada ou alterada são definidas em uma *Private Data Collection Policy*.
- Uma Private Data Collection pode ter tempo de vida.

Private Data Collections

- Caso o transação de um *chaincode* faça uso de PDC, o fluxo da transação possui etapa extra para geração de um TRANSACTION PROPOSAL RESPONSE.
- Transação possui leitura ou escrita em um PDC então:
- Gravação da informação privada gravada em base de *state* separada dentro do *CouchDB*.
- Replicação das informações privadas entre *peers* das organizações.
- PDC possui um Policy para definir as orgs que podem ter acesso a informação privada.
- Possibilidade de utilização de Transient Data para que os argumentos da transação não sejam gravados no bloco.
- *Transaction flow* de uma transação que usa PDC é mais complexo.

Exemplo de um channel com PDCs

- PDC1: Distributor, Farmer and Shipper
- PDC2: Distributor and Wholesaler
- PDC3: Wholesaler, Retailer and Shipper



PVC Policy (A or B; mínimo 2 peers)



ORGANIZAÇÃO A



Endorsing Peer

Método Chaincode $f(x)$

...

`PutPrivateData(asset)`



TransientData
World State
[asset]



ORGANIZAÇÃO B



Peer



TransientData
World State
[asset]

ORGANIZAÇÃO C



Peer



World State
[hash(asset)]

ORGANIZAÇÃO ORDERER



Exemplo de um asset em PDC

```
var Secret = assets.AssetType{
    Tag:      "secret",
    Label:    "Secret",
    Description: "Secret between Org2 and Org3",
    Readers: []string{"org2MSP", "org3MSP"},
    Props: []assets.AssetProp{
        {
            IsKey: true,
            Tag:   "secretName",
            Label: "Secret Name",
            DataType: "string",
            Writers: []string{"org2MSP"}, // This means only org2 can create the asset (org3 can edit)
        },
        {
            Tag: "secret",
            Label: "Secret",
            DataType: "string",
        },
    },
}
```

Arquivo collections2.json

Cada *asset PDC (Readers)* deve ter um elemento correspondente dentro do arquivo *collections2.json*

```
[
  {
    "name": "secret",
    "requiredPeerCount": 0,
    "maxPeerCount": 3,
    "blockToLive": 1000000,
    "memberOnlyRead": true,
    "policy": "OR('org2MSP.member', 'org3MSP.member')"
  }
]
```

D

Orquestradores Blockchain

Orquestradores Hyperledger Fabric

Estão disponíveis diversos orquestrados para facilitar a operação em redes Hyperledger Fabric

- Orquestradores open source
- Orquestradores em nuvem
- Orquestradores licenciados

Orquestradores open-source

Disponibilizados pela Hyperledger Foundation

Hyperledger Cello

<https://www.hyperledger.org/use/cello>

Hyperledger Bevel

<https://www.hyperledger.org/use/bevel>

Fabric Operator (Hyperledger Labs)

<https://labs.hyperledger.org/labs/hlf-operator.html>

Orquestradores em nuvem e licenciados

Disponíveis na nuvens públicas

Amazon Managed Blockchain

<https://aws.amazon.com/pt/managed-blockchain/>

IBM Blockchain Platform

<https://www.ibm.com/products/blockchain-platform-hyperledger-fabric>

Oracle Blockchain Platform Service

<https://www.oracle.com/br/blockchain/cloud-platform/>

Binario Cloud Blockchain (GoFabric engine)

<https://binario.cloud/produtos/blockchain>

GoFabric

Orquestrador disponibilizado pela empresa GoLedger

<https://gofabric.io/>

- Lista de redes Hyperledger Fabric
- Dashboard de rede
- Gestão multichannel
- Operações de escalabilidade da rede (*addPeer*, *AddOrg*, etc)
- Integração automática com chaincodes desenvolvidos com a biblioteca CC-Tools
- Atualização de chaincodes
- Atualização de APIs
- Mapeamento de dados dos chaincodes utilizando *Templates* de dados


GoFabric

Orquestrador disponibilizado pela empresa GoLedger

<https://gofabric.io/>

- Lista de redes Hyperledger Fabric
- Dashboard de rede
- Gestão multichannel
- Operações de escalabilidade da rede (*addPeer*, *AddOrg*, etc)
- Integração automática com chaincodes desenvolvidos com a biblioteca CC-Tools
- Atualização de chaincodes
- Atualização de APIs
- Mapeamento de dados dos chaincodes utilizando *Templates* de dados

GoFabric – Dashboard

GOFABRIC

DASHBOARD

NETWORK

NODES

CHAINCODES

API

SETUP MACHINES

?

🔔

🌐

MSARRES

M

GoFabric networks

✓ Networks loaded

digital-certificate-id-

cert-channel

id-channel

New network

DIGITAL-CERTIFICATE-ID-

CHANNELS

cert-channel

Orgs3Peers4Orderers5

id-channel

Orgs3Peers4Orderers5

ORGANIZATIONS

international-student-id.org

Peers2Orderers1APIs2

Joined Channels

- cert-channel
- id-channel

social-admin.gov

Peers1Orderers1APIs1

Joined Channels

- id-channel

CERT-CHANNEL

4

3

5

FABRIC VERSION

ACTIVE PEERS

ORGS

ORDERERS

2.2

CHAINCODE

VERSION

certificates

1.0

international-student-id.org

ca: 45.225.25.191

• orderer0: 45.225.25.191

• peer0: 45.225.25.191

• peer1: 45.225.25.240

ccapi:

none

45.225.25.240

ENTER GRAFANA

stanford-university.edu

ca: 45.225.25.74

• orderer0: 45.225.25.74

• peer0: 45.225.25.74

ccapi:

45.225.25.74

ENTER GRAFANA

ucla.edu

ca: 45.225.25.218

• orderer0: 45.225.25.218

• peer0: 45.225.25.218

ccapi:

45.225.25.218

GoFabric – Data Mapping Templates

GoFabric DASHBOARD NETWORK NODES CHAINCODES API SETUP MACHINES ? MSARRES M

GoFabric networks
✓ Networks loaded

digital-certificate-id-
cert-channel
id-channel
+ New network

TEMPLATES

TEMPLATE LIST

certificateTemplate		
certificates		
idTemplate		

EDIT TEMPLATE

Name
idTemplate

Description

Assets CREATE NEW

Label	Tag	
Passport	passport	

Fields

Label	Tag	
Passport ID		<input checked="" type="checkbox"/> Is key
id		<input checked="" type="checkbox"/> Required
Data type		<input type="checkbox"/> Read only
String		

Label
Name

Tag
name

☐ Is key

☒ Required

Number List of Number

String List of String

Boolean List of Boolean

GoFabric – deployment na nuvem



ORGANIZATIONS

international-student-id × social-admin × stanford-university × state-department × >

ADD MORE

ORGANIZATION BASIC INFO

Organization name: international-student-id Domain name: org

CA INFO

CA address: 45.225.25.191 CA user: admin

IP or domain address for CA: You must save this

CA password: Repeat password: Must be equal to CA password

INSTANCES

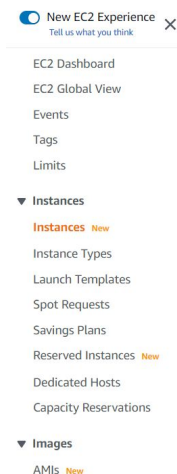
Machine configuration

Instance address: +

☒ Peer ☐ Orderer

Instance address: 45.225.25.191 ✓ Peer ✓ Orderer

Instance address: 45.225.25.240 ✓ Peer



Instances (3) info

Search

Instance state = running X Clear filters

<input type="checkbox"/>	Name	Instance ID	Instance state	Public IPv4 ...
<input type="checkbox"/>	-	i-0009f0a932eddf9c5	Running	3.129.44.196
<input type="checkbox"/>	goprocess-audit	i-02cdb10f594c407be	Running	3.135.62.89
<input type="checkbox"/>	goprocess-regi...	i-077650e507dbb8c94	Running	18.221.179.150

Select an instance

GoFabric – multichannel

GOFABRIC

DASHBOARD NETWORK NODES CHAINCODES API SETUP MACHINES ? MSARRES M

GoFabric networks
✓ Networks loaded

digital-certificate-id- ▾

cert-channel

id-channel

+ New network

Define Organizations ✓ 2 Define Channels 3 Define Chaincodes 4 Start network

CHANNELS

cert-channel x id-channel x

ADD CHANNEL

CHANNEL BASIC INFO

Channel name
cert-channel


Peers Available

- international-student-id
- social-admin
- 45.225.25.120 +
- stanford-university
- state-department

Peers in Channel

- international-student-id
- 45.225.25.191 ✗
- 45.225.25.240 ✗
- stanford-university
- 45.225.25.74 ✗

GoFabric – Chaincode marketplace

 **GOFABRIC**

GoFabric networks
✓ Networks loaded

digital-certificate-id- ▾

cert-channel

id-channel

+ New network

cert-channel id-channel

certificates ×

ADD CHAINCODE

Chaincode name
certificates

TEMPLATE

CLOUD CHAINCODES

CHAINCODE FILE

Select the template chaincode ⓘ

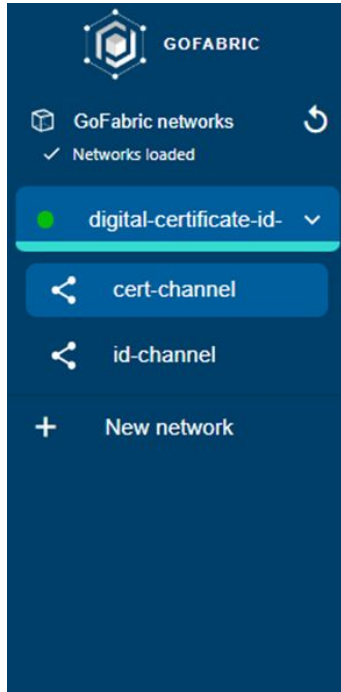
GOSHARE

Select the user template chaincode ⓘ

GOBIO

GOPROCESS

GoFabric – Private Chaincode



PERMISSIONS

i If no permissions are set, every organization in the network will be able to read and write to every asset

Passport

Asset

international-student-id, stanford-university

☒ Private data **i**

☒ international-student-id

☐ social-admin

☒ stanford-university

☐ state-department

☐ ucla

Private data

Passport ID string

Prop

All enabled

GoFabric – Atualizar Chaincode

The screenshot displays the GoFabric web application interface. The top navigation bar includes links for DASHBOARD, NETWORK, NODES, CHAINCODES, API, and SETUP MACHINES. The left sidebar shows the network hierarchy: GoFabric networks, Networks loaded, digital-certificate-id (selected), cert-channel (selected), id-channel, and New network.

UPGRADE CHAINCODE

certificates

Chaincode version: 2.0

☒ AUTO VERSION UPGRADE

CHANNEL PEERS UPGRADE

- international-student-id
 - ☒ peer0.international-student-id.org - 45.225.25.191
 - ☒ peer1.international-student-id.org - 45.225.25.240
- stanford-university
 - ☐ ...

Select the template chaincode ⓘ

GOSHARE

Select the user template chaincode ⓘ

GOBIO **GOPROCESS**

Add your template

certificates

PERMISSIONS

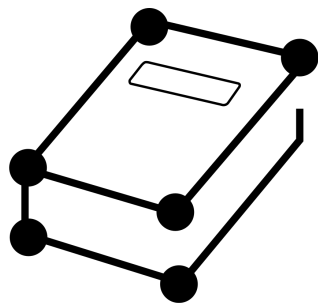
ⓘ If no permissions are set, every organization in the network will be able to read and write to every asset

Document Hash: All

Asset: Private data

Course: All

Private



GoLedger

Marcos Sarres

CEO

marcos.sarres@goledger.com.br



goledger.com.br



hello@goledger.com.br



<https://medium.com/@goledger>



<http://linkedin.com/company/goledger>



<https://www.youtube.com/goledger>