# Network Setup and Testing

Prerequisites

Ubuntu

To run Hyperledger Composer and Hyperledger Fabric, you should have at least 4Gb of memory.

The following are prerequisites for installing the required development tools:

* Operating Systems: Ubuntu Linux 14.04 / 16.04 LTS (both 64-bit), or Mac OS 10.12
* Docker Engine: Version 17.03 or higher
* Docker-Compose: Version 1.8 or higher
* Node: 8.9 or higher (note version 9 is not supported)
* npm: v5.x
* git: 2.9.x or higher
* Python: 2.7.x
* A code editor of your choice, we recommend VSCode.

\*\*If installing Hyperledger Composer using Linux, be aware of the following advice:

* Login as a normal user, rather than root.
* Do not su to root.
* When installing prerequisites, use curl, then unzip using sudo.
* Run prereqs-ubuntu.sh as a normal user. It may prompt for root password as some of it's actions are required to be run as root.
* Do not use npm with sudo or su to root to use it.
* Avoid installing node globally as root.\*\*

If you're running on Ubuntu, you can download the prerequisites using the following commands:

curl -O https://hyperledger.github.io/composer/latest/prereqs-ubuntu.sh

chmod u+x prereqs-ubuntu.sh

Next run the script - as this briefly uses sudo during its execution, you will be prompted for your password.

./prereqs-ubuntu.sh

# Installing components

### **Step 1: Install the CLI tools**

There are a few useful CLI tools for Composer developers. The most important one is composer-cli, which contains all the essential operations, so we'll install that first. Next, we'll also pick up generator-hyperledger-composer, composer-rest-server and Yeoman plus the generator-hyperledger-composer.

Essential CLI tools:

npm install -g composer-cli

1. Utility for running a REST Server on your machine to expose your business networks as RESTful APIs:

npm install -g composer-rest-server

1. Useful utility for generating application assets:

npm install -g generator-hyperledger-composer

1. Yeoman is a tool for generating applications, which utilises generator-hyperledger-composer:

npm install -g yo

### **Step 2: Install Playground**

1. Browser app for simple editing and testing Business Networks:

npm install -g composer-playground

### **Step 3: Set up your IDE**

Whilst the browser app can be used to work on your Business Network code, most users will prefer to work in an IDE. Our favourite is VSCode, because a Composer extension is available.

1. Install VSCode from this URL: <https://code.visualstudio.com/download>
2. Open VSCode, go to Extensions, then search for and install the Hyperledger Composer extension from the Marketplace.

### **Step 4: Install Hyperledger Fabric**

This step gives you a local Hyperledger Fabric runtime to deploy your business networks to.

1. In a directory of your choice (we will assume ~/fabric-dev-servers), get the .tar.gz file that contains the tools to install Hyperledger Fabric:

mkdir ~/fabric-dev-servers && cd ~/fabric-dev-servers

curl -O https://raw.githubusercontent.com/hyperledger/composer-tools/master/packages/fabric-dev-servers/fabric-dev-servers.tar.gz

tar -xvf fabric-dev-servers.tar.gz

A zip is also available if you prefer: just replace the .tar.gz file with fabric-dev-servers.zip and the tar -xvf command with a unzip command in the preceding snippet.

1. Use the scripts you just downloaded and extracted to download a local Hyperledger Fabric runtime:

cd ~/fabric-dev-servers

./downloadFabric.sh

# Controlling your dev environment

## **Starting and stopping Hyperledger Fabric**

You control your runtime using a set of scripts which you'll find in ~/fabric-dev-servers if you followed the suggested defaults.

The first time you start up a new runtime, you'll need to run the start script, then generate a PeerAdmin card:

cd ~/fabric-dev-servers

./startFabric.sh

./createPeerAdminCard.sh

You can start and stop your runtime using ~/fabric-dev-servers/stopFabric.sh, and start it again with ~/fabric-dev-servers/startFabric.sh.

At the end of your development session, you run ~/fabric-dev-servers/stopFabric.sh and then ~/fabric-dev-servers/teardownFabric.sh. Note that if you've run the teardown script, the next time you start the runtime, you'll need to create a new PeerAdmin card just like you did on first time startup.

The local runtime is intended to be frequently started, stopped and torn down, for development use. If you're looking for a runtime with more persistent state, you'll want to run one outside of the dev environment, and deploy Business Networks to it. Examples of this include running it via Kubernetes, or on a managed platform such as IBM Cloud.

## **Start the web app ("Playground")**

To start the web app, run:

composer-playground

It will typically open your browser automatically, at the following address: <http://localhost:8080/login>

You should see the PeerAdmin@hlfv1 Card you created with the createPeerAdminCard script on your "My Business Networks" screen in the web app: if you don't see this, you may not have correctly started up your runtime!

# Destroy a previous setup

If you've previously used an older version of **Hyperledger Composer** and are now setting up a new install, you may want to kill and remove all previous Docker containers, which you can do with these commands:

docker kill $(docker ps -q)

docker rm $(docker ps -aq)

docker rmi $(docker images dev-\* -q)

# Interacting with other business networks

Hyperledger Composer includes functionality that can be used by a business network to access an asset, participant, or transaction that is recorded in another business network.

## **Step One: Starting a Hyperledger Fabric network**

You must start a Hyperledger Fabric network. You can use the simple Hyperledger Fabric network provided in the development environment, or you can use your own Hyperledger Fabric network that you have built by following the Hyperledger Fabric documentation.

We assume that you use the simple Hyperledger Fabric network provided in the development environment. If you use your own Hyperledger Fabric network, then you must map between the configuration detailed below and your own configuration.

1. Start a clean Hyperledger Fabric by running the following commands:

cd ~/fabric-dev-servers

./stopFabric.sh

./teardownFabric.sh

./downloadFabric.sh

./startFabric.sh

1. Delete any business network cards that may exist in your wallet. It is safe to ignore any errors that state that the business network cards cannot be found:

composer card delete -c PeerAdmin@hlfv1

If these commands fail, then you have business network cards from a previous version and you will have to delete the file system card store.

rm -fr ~/.composer

1. Create the Peer Admin Card by running the following command

./createPeerAdminCard.sh

## **Step Two: Define the business networks**

## **Creating business network structure hc**

The key concept for Hyperledger Composer is the **business network definition (BND)**. It defines the data model, transaction logic and access control rules for your blockchain solution. To create a BND, we need to create a suitable project structure on disk.

The easiest way to get started is to use the Yeoman generator to create a skeleton business network. This will create a directory containing all of the components of a business network.

1. Create a skeleton business network using Yeoman. This command will require a business network name, description, author name, author email address, license selection and namespace.

yo hyperledger-composer:businessnetwork

1. Enter hc for the network name, and desired information for description, author name, and author email.
2. Select Apache-2.0 as the license.
3. Select org.cs as the namespace.
4. Select No when asked whether to generate an empty network or not.

## **Defining business network hc**

A business network is made up of assets, participants, transactions, access control rules, and optionally events and queries. In the skeleton business network created in the previous steps, there is a model (.cto) file which will contain the class definitions for all assets, participants, and transactions in the business network. The skeleton business network also contains an access control (permissions.acl) document with basic access control rules, a script (logic.js) file containing transaction processor functions, and a package.json file containing business network metadata.

#### Modelling assets, participants, and transactions

The first document to update is the model (.cto) file. This file is written using the [Hyperledger Composer Modelling Language](https://hyperledger.github.io/composer/latest/reference/cto_language.html). The model file contains the definitions of each class of asset, transaction, participant, and event. It implicitly extends the Hyperledger Composer System Model described in the modelling language documentation.

1. Open the org.cs.cto model file.
2. Replace the contents with the following:

/\*\*

\* My commodity trading network

\*/

namespace org.cs  
  
asset MedicalIssue identified by issueID {  
    o String issueID  
    o String description  
    o String docSelected  
    o Double bill  
    o String docSuggested  
    --> Patient reported  
}  
  
participant Patient identified by patientId {  
    o String patientId  
    o String name  
    o String policyNo  
}  
transaction Advice {  
    --> MedicalIssue medicalIssue  
    --> Patient newReported  
}

1. Save your changes to org.cs.cto.

#### Adding JavaScript transaction logic

In the model file,  Advice transaction was defined, specifying a relationship to an asset, and a participant. The transaction processor function file contains the JavaScript logic to execute the transactions defined in the model file.

The Advice transaction is intended to simply accept the identifier of the MedicalIssue asset which is being reported, and the identifier of the Patient participant to set as the new owner.

1. Open the logic.js script file.
2. Replace the contents with the following:

/\*\*

\* Track the trade of a commodity from one trader to another

\* @param {org.cs.Advice} advice  - the trade to be processed

\* @transaction

\*/

async function patientReport(advice) {  
  advice.medicalIssue.reported = advice.newReported;  
  
  const otherNetworkData = await getNativeAPI().invokeChaincode('hcg', ['getResourceInRegistry', 'Asset', 'org.cs.MedicalIssue', advice.medicalIssue.issueID], 'composerchannel');                      
  const stringAsset = new Buffer(otherNetworkData.payload.toArrayBuffer()).toString('utf8');  
  const asset = getSerializer().fromJSON(JSON.parse(stringAsset));  
  
  advice.medicalIssue.docSuggested = asset.docSuggested;  
    
  
  const assetRegistry = await getAssetRegistry('org.cs.MedicalIssue');  
  await assetRegistry.update(advice.medicalIssue);  
}

1. Save your changes to logic.js.

#### Adding access control

1. Replace the following access control rules in the file permissions.acl :

/\*\*

\* Access control rules for tutorial-network

\*/

rule Default {

description: "Allow all participants access to all resources"

participant: "ANY"

operation: ALL

resource: "org.cs.\*"

action: ALLOW

}

rule SystemACL {

description: "System ACL to permit all access"

participant: "ANY"

operation: ALL

resource: "org.hyperledger.composer.system.\*\*"

action: ALLOW

}

1. Save your changes to permissions.acl.

Follow above two create a business network called hcg. This will be network B.

## **Step One: Creating business network structure hcg**

The key concept for Hyperledger Composer is the **business network definition (BND)**. It defines the data model, transaction logic and access control rules for your blockchain solution. To create a BND, we need to create a suitable project structure on disk.

The easiest way to get started is to use the Yeoman generator to create a skeleton business network. This will create a directory containing all of the components of a business network.

1. Create a skeleton business network using Yeoman. This command will require a business network name, description, author name, author email address, license selection and namespace.

yo hyperledger-composer:businessnetwork

1. Enter hcg for the network name, and desired information for description, author name, and author email.
2. Select Apache-2.0 as the license.
3. Select org.cs as the namespace.
4. Select No when asked whether to generate an empty network or not.

## **Defining business network hcg**

1. Open the org.cs.cto model file.
2. Replace the contents with the following:

/\*\*

\* My commodity trading network

\*/

namespace org.cs  
  
  
asset MedicalIssue identified by issueID {  
    o String issueID  
    o String description  
    o String docSelected  
    o Double bill  
    o String docSuggested  
    --> Patient reported  
}  
  
participant Consultancy identified by consultancyID {  
    o String consultancyID  
    o String docId  
    o String treats  
    o String fee  
    o String coveredInsurance  
}  
  
participant Patient identified by patientId {  
    o String patientId  
    o String name  
    o String policyNo  
}  
transaction Advice {  
    --> MedicalIssue medicalIssue  
    --> Patient newReported  
    --> Consultancy consultancy  
} Save your changes to

#### Adding JavaScript transaction logic

In the model file, a Advice transaction was defined, specifying a relationship to an asset, and a participant. The transaction processor function file contains the JavaScript logic to execute the transactions defined in the model file.

The Advice transaction is intended to simply accept the identifier of the Medicalissue asset which is being updated, and the identifier of the Patient and Consultancy participant to set as the new patient and advised consultant respectively.

1. Open the logic.js script file.
2. Replace the contents with the following:

/\*\*

\* Track the trade of a commodity from one trader to another

\* @param {org.cs.Advice} advice - the trade to be processed

\* @transaction

\*/

async function tradeCommodity(advice) {  
    advice.medicalIssue.docSuggested = advice.consultancy.docId;  
 advice.medicalIssue.reported = advice.newReported;  
    let assetRegistry = await getAssetRegistry('org.cs.MedicalIssue');  
    await assetRegistry.update(advice.medicalIssue);  
}

1. Save your changes to logic.js.

#### Adding access control

1. Replace the following access control rules in the file permissions.acl :

/\*\*

\* Access control rules for tutorial-network

\*/

rule Default {

description: "Allow all participants access to all resources"

participant: "ANY"

operation: ALL

resource: "org.cs.\*"

action: ALLOW

}

rule SystemACL {

description: "System ACL to permit all access"

participant: "ANY"

operation: ALL

resource: "org.hyperledger.composer.system.\*\*"

action: ALLOW

}

1. Save your changes to permissions.acl.

## **Step Three: Generate a business network archive**

Now that the business network has been defined, it must be packaged into a deployable business network archive (.bna) file.

1. Using the command line, navigate to the hc directory.
2. From the hc directory, run the following command:

composer archive create -t dir -n .

After the command has run, a business network archive file called hc@0.0.1.bna has been created in the hc directory.

## **Step Three: Deploy the business networks**

After creating the .bna file, the business network can be deployed to the instance of Hyperledger Fabric. Normally, information from the Fabric administrator is required to create a PeerAdmin identity, with privileges to install chaincode to the peer as well as start chaincode on the composerchannel channel. However, as part of the development environment installation, a PeerAdmin identity has been created already.

After the business network has been installed, the network can be started. For best practice, a new identity should be created to administer the business network after deployment. This identity is referred to as a network admin.

#### Retrieving the correct credentials

A PeerAdmin business network card with the correct credentials is already created as part of development environment installation.

#### Deploying the business network

Deploying a business network to the Hyperledger Fabric requires the Hyperledger Composer business network to be installed on the peer, then the business network can be started, and a new participant, identity, and associated card must be created to be the network administrator. Finally, the network administrator business network card must be imported for use, and the network can then be pinged to check it is responding.

1. Install and start business network A using the following commands

composer network install --card PeerAdmin@hlfv1 --archiveFile hc@0.0.1.bna

composer network start --networkName hc --networkVersion 0.0.1 --networkAdmin admin --networkAdminEnrollSecret adminpw --card PeerAdmin@hlfv1 --file networkA.card

composer card import --file networkA.card --card networkA

1. Install and start business network B using the following commands

composer network install --card PeerAdmin@hlfv1 --archiveFile hcg@0.0.1.bna

composer network start --networkName hcg --networkVersion 0.0.1 --networkAdmin admin --networkAdminEnrollSecret adminpw --card PeerAdmin@hlfv1 --file networkB.card

composer card import --file networkB.card --card networkB

1. To check that the business networks have been deployed successfully run the following commands to ping the business networks

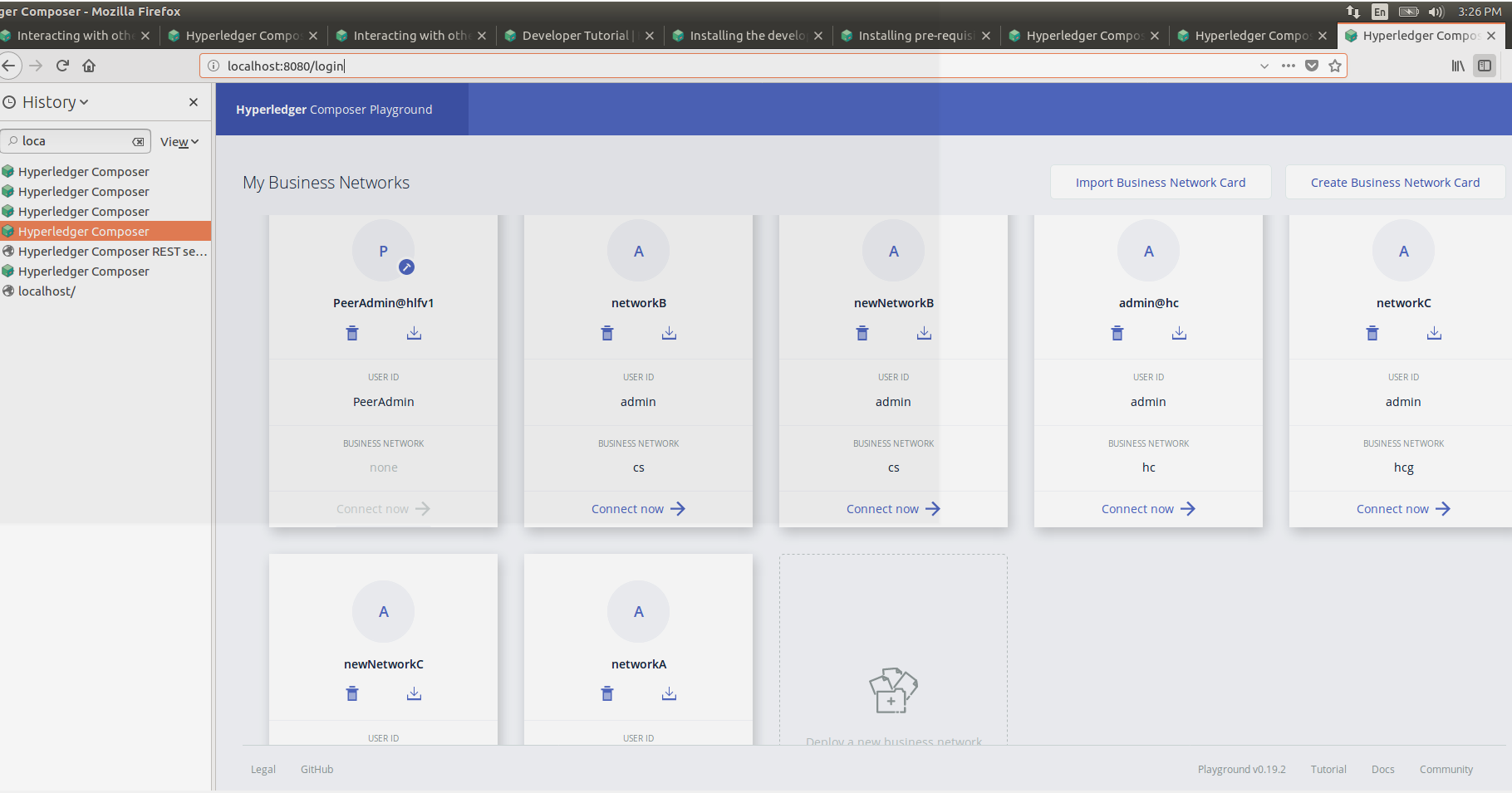
composer network ping --card networkA

composer network ping --card networkB

## **Step Four: Create the assets**

Open localhost:8080/login

You can view all business network created



1. Create a participant in business network A test tab

A screenshot of a computer

Description generated with very high confidence

1. Create an asset in business network A

A screenshot of a computer

Description generated with very high confidence

1. Create participants in business network B. Run the following command.

A screenshot of a computer

Description generated with very high confidence

A screenshot of a computer

Description generated with very high confidence

1. Create an asset in business network B.

A screenshot of a computer

Description generated with very high confidence

## **Step Five: Bind the identity on network A to the participant on network B**

1. Export the networkA card to get the credentials

composer card export -c networkA

1. Unzip the card, you may need to rename networkA.card to networkA.zip.
2. Bind the identity to the participant. Run the following command.

composer identity bind --card networkB --participantId resource:org.hyperledger.composer.system.NetworkAdmin#admin --certificateFile ./networkA/credentials/certificate

1. Create a card with the bound identity.

composer card create -p ~/.composer/cards/networkB/connection.json --businessNetworkName hcg -u admin -c ./networkA/credentials/certificate -k ./networkA/credentials/privateKey -f newNetworkB.card

1. Import the card

composer card import --file newNetworkB.card --card newNetworkB

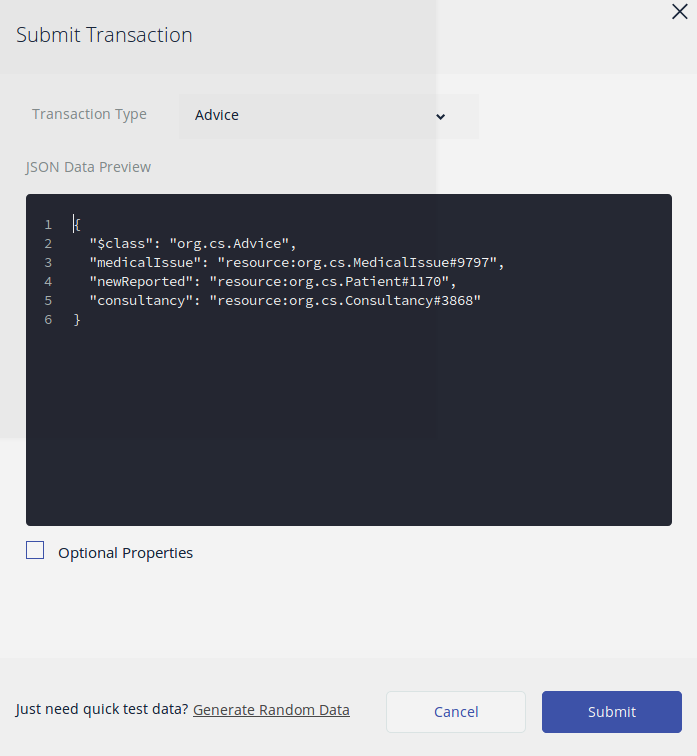
1. Ping the network to activate the identity

composer network ping --card newNetworkB

## **Step Six: Submit a transaction**

Submit a transaction to see the effect of querying an asset on a different business network.

On network B



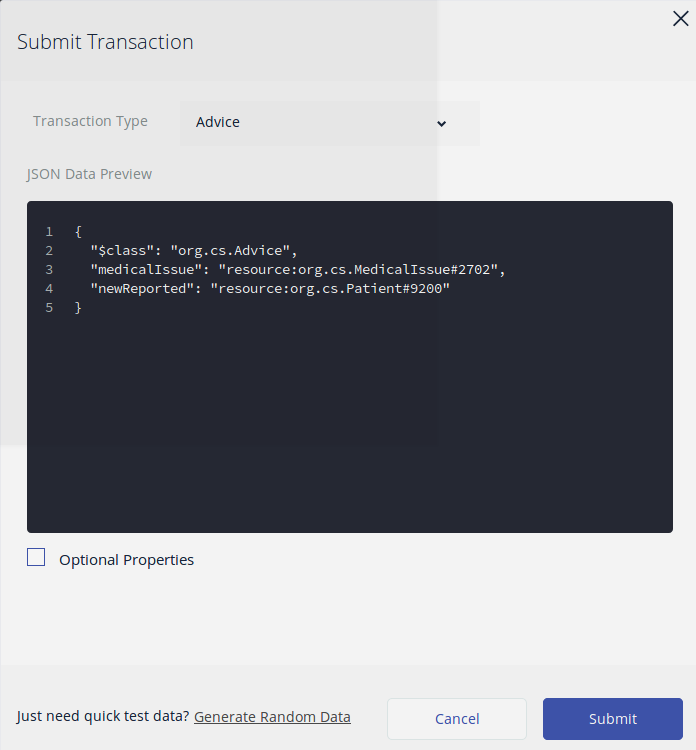
**ep Eight: Check the updated asset**

View the updated asset to check that the docSuggested was correctly updated.

A screenshot of a computer

Description generated with very high confidence

On network A



See that patient can view suggested doctor Ag