IBM Blockchain Platform

Tutorial A4: Invoking a smart contract from VS Code

Estimated time: `15 minutes`

In the last tutorial we packaged and deployed our smart contract on a local Hyperledger Fabric network. In this tutorial we will:

- Learn about identities, wallets and gateways
- Exercise a smart contract directly from VS Code
- Understand the difference between evaluating and submitting transactions

In order to successfully complete this tutorial, you must have first completed tutorial A3: Deploying a smart contract in the active VS Code workspace.

▢ **A4.1**: Expand the first section below to get started.

▶ **Connect to the Hyperledger Fabric gateway**

In order to submit transactions in Hyperledger Fabric you will need an identity, a wallet and a gateway.

Identities, wallets and gateways

The resources that you can access in a Hyperledger Fabric network are determined according to your identity; that's why its called a permissioned blockchain. Your identity is typically represented by an X.509 certificate issued by your organization, and stored in your wallet. Once you have an identity and a wallet, you can create a gateway that allows you to submit transactions to a network.

A gateway represents a connection to a set of Hyperledger Fabric networks. If you want to submit a transaction, whether using VS Code or your own application, a gateway makes it easy to interact with a network: you configure a gateway, connect to it with an identity from your wallet, choose a particular network, and start submitting transactions using a smart contract that has been deployed in that network.

A gateway is configured using a connection profile, which identifies a single peer in the network as an initial connection point. We're going to use a pre-configured gateway that was created when we started the one organization network.

> **Want to know more about gateways?** Read about them in the Hyperledger Fabric documentation.

Take care to understand the difference between **Fabric Environments** and **Fabric Gateways**: an *environment* gives an overview of all the resources available to you in a Hyperledger Fabric network; a *gateway* provides an access point to those resources.

Gateways and Wallets in VS Code

When the one organization network was created in the previous tutorial, a gateway was created for you at the same time; this is now shown in the Fabric Gateways view. This view allows you to add new gateways to submit transactions to both local and remote Hyperledger Fabric networks.
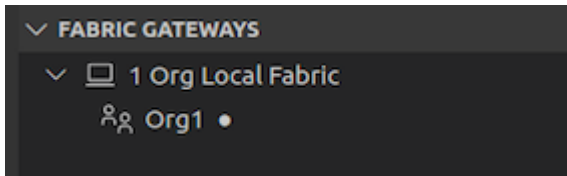
Furthermore, every component in this network has an assigned identity, and these are shown in Fabric Wallets view. In this view you can see two wallets, each containing two identities:
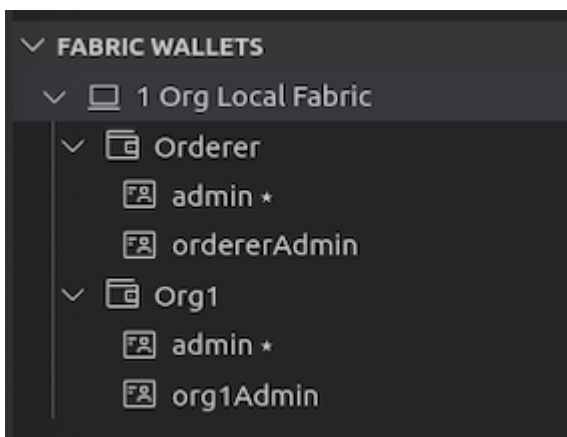
These identities are used as follows:

- Orderer wallet
    - *admin*: this identity is associated with the Orderer node
    - *ordererAdmin*: this identity can be used to interact with the ordering service
- Org1 wallet
    - *admin*: this identity is associated with the Org1Peer1 node
   - *org1Admin*: this identity will be used to submit and evaluate transactions

It is the *org1Admin* identity that we will be using to exercise the smart contract.

Connecting to the Fabric Gateway in VS Code

We will now connect to a gateway using the *org1Admin* identity.

▢   A4.2:   In the Fabric Gateways view, click "Org1".

If you can't see this view, remember to first click the IBM Blockchain Platform icon in the activity bar.
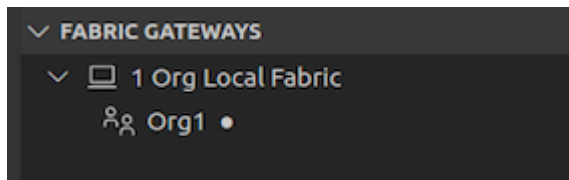
You will now be asked to select the identity that the gateway will use to connect to the network. As a reminder, we will be using the identity *org1Admin*.

⬜ **A4.3:** Click 'org1Admin'.
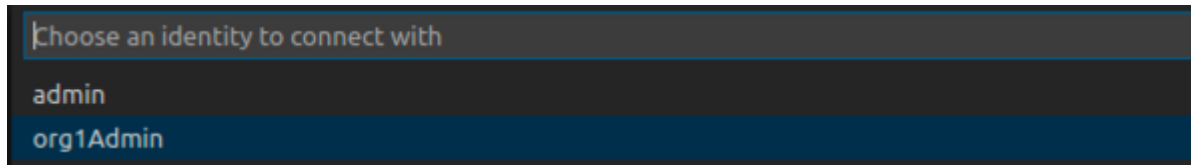
The IBM Blockchain Platform VS Code extension will now connect to the local Hyperledger Fabric gateway; this will only take a few seconds to complete.

Once connected, notice that the view changes to reflect the channels, smart contracts and transactions available to the connected gateway.

⬜ **A4.4:** Fully expand the Channels tree in the Fabric Gateways view to show the available transactions.
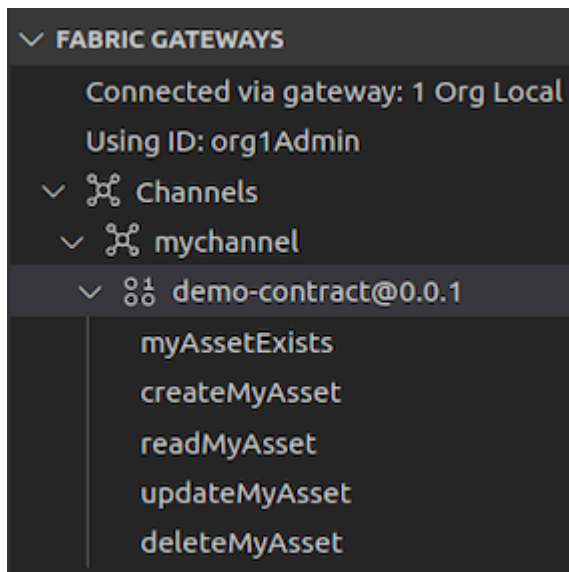
The tree shows the *mychannel* network into which the smart contract was deployed, the *demo-contract* smart contract that was deployed (including its version), and the five transaction methods that are available to applications.

⬜ **A4.5:** Expand the next section of the tutorial to continue.

▶ **Invoke transactions using the gateway**

We will now exercise your smart contract. Hyperledger Fabric can generate two different kinds of transactions:

- *Submitted* transactions are recorded on the blockchain ledger. These are used when you want to update the current value of the ledger. Submitted transactions go through the full consensus process before they are recorded on the ledger. It is possible to submit read-only ledger transactions, but it's less common.

- *Evaluated* transactions are not recorded on the blockchain ledger. These transactions are typically used when you want to simply query the current value of the ledger. Evaluated transactions do not go through the consensus process; they are run on a single peer, and the result is returned to the caller. It is possible to evaluate read-write transactions, but it's less common.

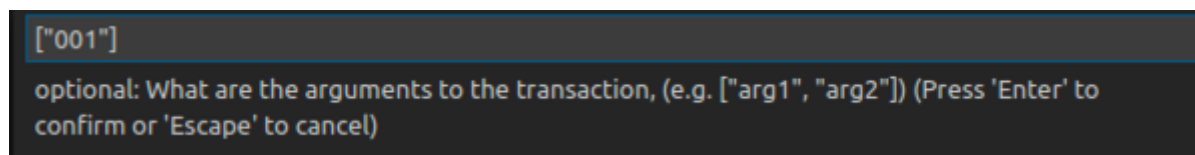The VS Code extension allows you to both submit and evaluate transactions.

We will start by evaluating the transaction 'myAssetExists'.

☐ **A4.6**: Right-click 'myAssetExists' and select 'Evaluate Transaction'.
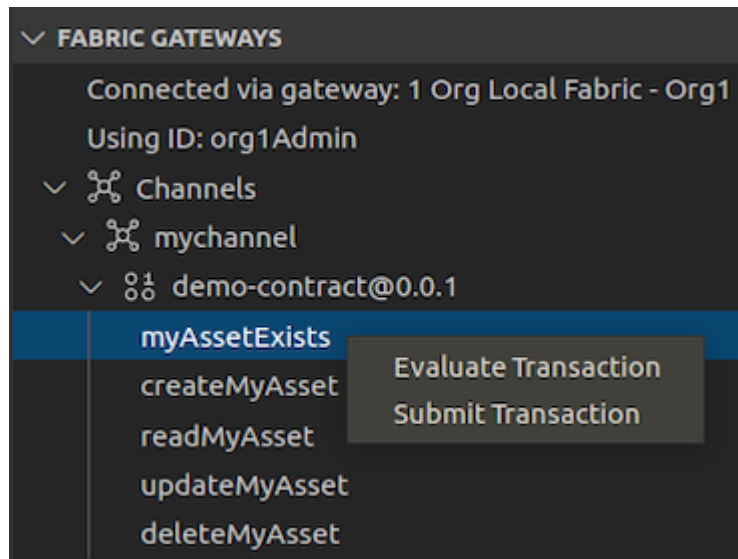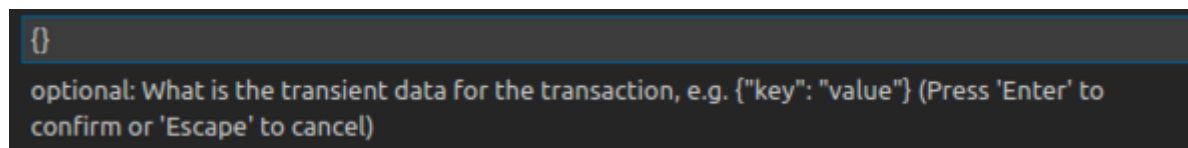

Transaction inputs are entered using JSON.

☐ **A4.7**: Replace the input parameters with `["001"]` and press Enter.

```
["001"]
```
optional: What are the arguments to the transaction, (e.g. ["arg1", "arg2"]) (Press 'Enter' to confirm or 'Escape' to cancel)

Transient data is an advanced feature that we will cover in a later tutorial; we will not use it here.

☐ **A4.8**: Press Enter again to accept the default transient data parameters.

```
{}
```
optional: What is the transient data for the transaction, e.g. {"key": "value"} (Press 'Enter' to confirm or 'Escape' to cancel)

The demo-contract smart contract will now run on the peer to generate a myAssetExists transaction response using the transaction input "001". As you will recall from the smart contract code, the contract will return true if the business object with key "001" exists in the state database, or false otherwise.

When the method completes, the Output tab will show the results of the evaluation.

☐ **A4.9**: Move the mouse over the "Successfully evaluated transaction" notification to reveal the close icon, and click it to close it.

If you review the output from the command you can see the results of evaluating the transaction. In this case, the return value is false, because the key does not exist in the state database.

We will now create the business object with key "001". This time, we will add a new transaction to the ledger, so we need to submit a transaction rather than evaluate one.

▢　**A4.10**:　Right-click 'createMyAsset' and select 'Submit Transaction'.
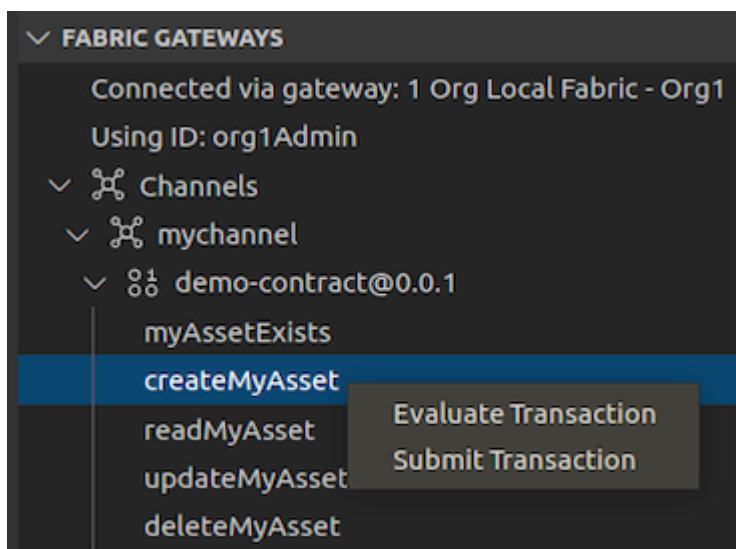
As you may recall, the createMyAsset transaction takes two parameters: a key and its associated value.

▢　**A4.11**:　Replace the input parameters with `["001","Mona Lisa"]` and press Enter.

▢　**A4.12**:　Press Enter a second time to accept the transient data defaults and submit the transaction.

Review the output to ensure that the transaction was successful.

```
DEBUG CONSOLE   PROBLEMS   OUTPUT   TERMINAL                                    Blockchain        ∨   ☰

[4/24/2020 1:22:54 PM] [INFO] submitTransaction
[4/24/2020 1:23:56 PM] [INFO] submitting transaction createMyAsset with args 001,Mona Lisa on channel mychannel
[4/24/2020 1:23:59 PM] [SUCCESS] No value returned from createMyAsset
```

🔲   **A4.13**:   Evaluate the "myAssetExists" transaction a second time with the "001" key to show that the asset now exists.

```
DEBUG CONSOLE   PROBLEMS   OUTPUT   TERMINAL                                    Blockchain        ∨

[4/24/2020 1:25:18 PM] [INFO] evaluateTransaction
[4/24/2020 1:25:26 PM] [INFO] evaluating transaction myAssetExists with args 001 on channel mychannel
[4/24/2020 1:25:26 PM] [SUCCESS] Returned value from myAssetExists: true
```

The returned value is now 'true'.

🔲   **A4.14**:   Submit the "updateMyAsset" transaction to change the value of the "001" key to "The Hay Wain".

```
DEBUG CONSOLE   PROBLEMS   OUTPUT   TERMINAL                                    Blockchain        ∨   ☰   🔒

[4/24/2020 1:26:42 PM] [INFO] submitTransaction
[4/24/2020 1:26:54 PM] [INFO] submitting transaction updateMyAsset with args 001,The Hay Wain on channel mychannel
[4/24/2020 1:26:56 PM] [SUCCESS] No value returned from updateMyAsset
```

🔲   **A4.15**:   Evaluate the "readMyAsset" transaction to return the updated value of the "001" key.

```
DEBUG CONSOLE   PROBLEMS   OUTPUT   TERMINAL                                    Blockchain        ∨

[4/24/2020 1:27:39 PM] [INFO] evaluateTransaction
[4/24/2020 1:27:43 PM] [INFO] evaluating transaction readMyAsset with args 001 on channel mychannel
[4/24/2020 1:27:43 PM] [SUCCESS] Returned value from readMyAsset: {"value":"The Hay Wain"}
```

🔲   **A4.16**:   Finally, submit the "deleteMyAsset" transaction to delete the "001" asset from the world state.

```
DEBUG CONSOLE   PROBLEMS   OUTPUT   TERMINAL                                    Blockchain        ∨

[4/24/2020 1:28:23 PM] [INFO] submitTransaction
[4/24/2020 1:28:28 PM] [INFO] submitting transaction deleteMyAsset with args 001 on channel mychannel
[4/24/2020 1:28:30 PM] [SUCCESS] No value returned from deleteMyAsset
```

Note carefully this last transaction! We have added a delete transaction to the blockchain, which has resulted in an empty state database for key "001". It is perfectly possible to delete assets from the world state, but submitted transactions are always added to the ledger. The blockchain records the changes that have happened to the world state database, which can include deleting records as well as adding and modifying them.

Summary

In this tutorial we have used identities, wallets and gateways to submit and evaluate smart contract transactions using the local default Hyperledger Fabric network provided with VS Code.

In the next tutorial we will build and use a standalone application to transact with the blockchain.