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.

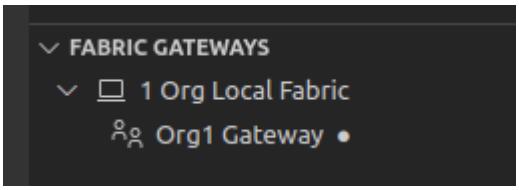
A gateway is configured using a connection profile, which identifies a single peer in the network as an initial connection point. We're going to use a pre-configured gateway that was created when we started the one organization network.

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

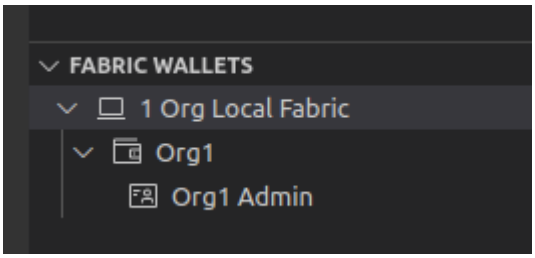
Take care to understand the difference between **Fabric Environments** and **Fabric Gateways**: an *environment* gives an overview of all the resources available to you in a Hyperledger Fabric network; a *gateway* provides an access point to those resources.

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, an *Org1* wallet with an identity *Org1 Admin* has been created in the Fabric Wallets view.



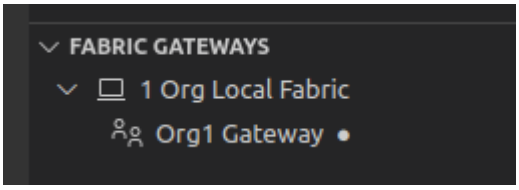
The Org1 Admin identity will be used in this tutorial to submit and evaluate transactions through the smart contract. In production and in more complex testing scenarios other non-admin identities would be used for the transactions.

Connecting to the Fabric Gateway in VS Code

We will now connect to a gateway using the *Org1 Admin* identity.

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

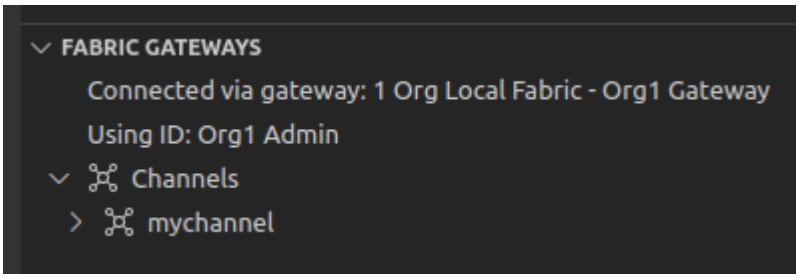
If you can't see this view, remember to first click the IBM Blockchain Platform icon in the activity bar.



There is only 1 identity in the Org1 wallet, so the IBM Blockchain Platform VS Code extension will use that identity automatically and connect to the local Hyperledger Fabric gateway; this will only take a few seconds to complete.

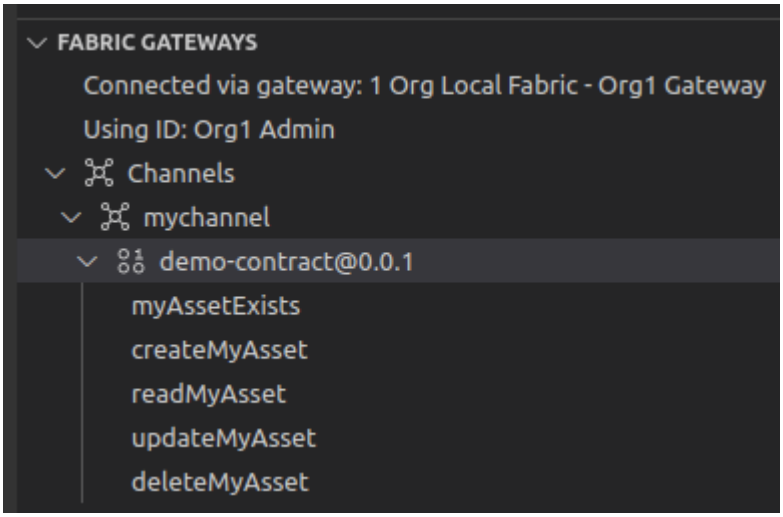
Once connected, notice that the view changes to reflect the channels available to the connected gateway.

A4.3: Review the channels.



The connected gateway shows the list of channels on the network, in this case just the single *mychannel*.

A4.4: Fully expand the Channels tree in the Fabric Gateways view to show the available transactions.



The expanded tree shows 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**

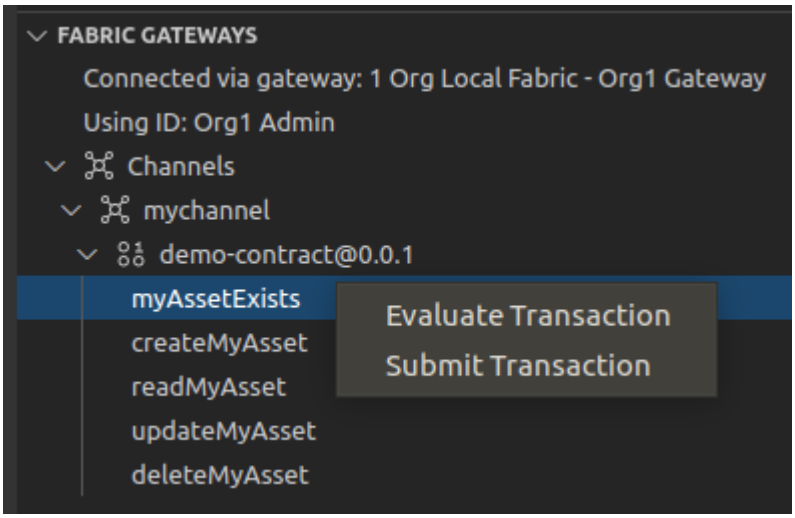
We will now exercise your smart contract. Hyperledger Fabric can generate two different kinds of transactions:

- *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.

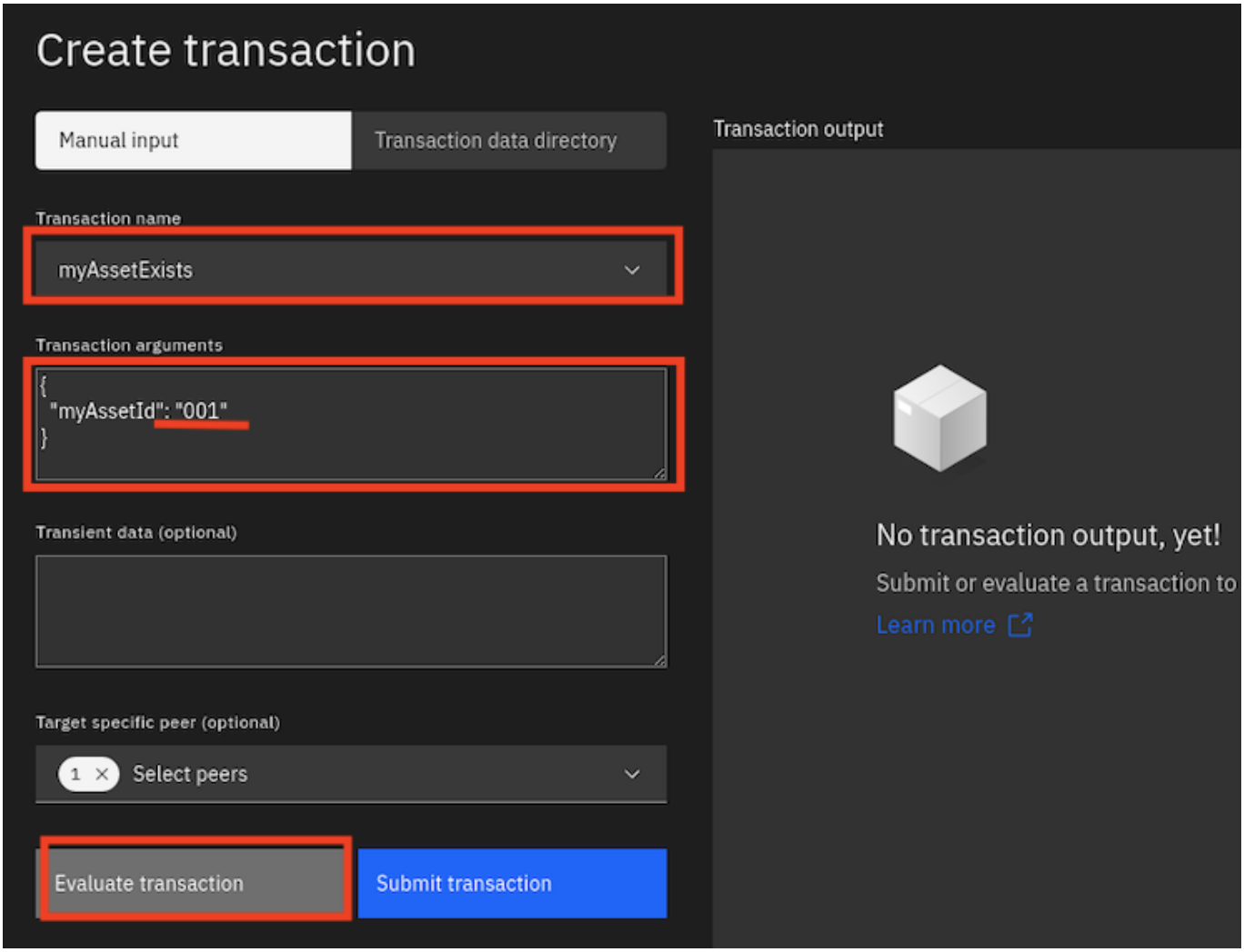
The VS Code extension allows you to both submit and evaluate transactions.

We will start by evaluating the transaction 'myAssetExists'.

A4.6: Click 'myAssetExists' which will open the Transaction View.



A4.7: The Transaction View allows the transaction to be prepared before being submitted or evaluated.



On the Transaction View, the *Transaction name* can be selected from the dropdown list and the *Transaction arguments* supplied. Transient data is an advanced feature that we will cover in a later tutorial and can be ignored. We have only 1 peer in the local fabric so target specific peer can be ignored too.

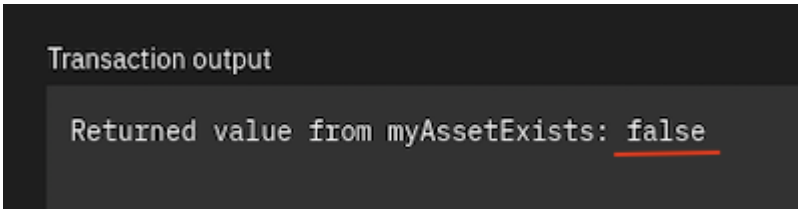
Ensure that *myAssetExists* is selected as the Transaction name, specify `"001"` for myAssetId in the JSON object pair.

Now click *Evaluate transaction*

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.

The Transaction view will be used several times in this tutorial and if it is closed or hidden at any point, just click any of the transactions in the Fabric Gateways view to bring it back.

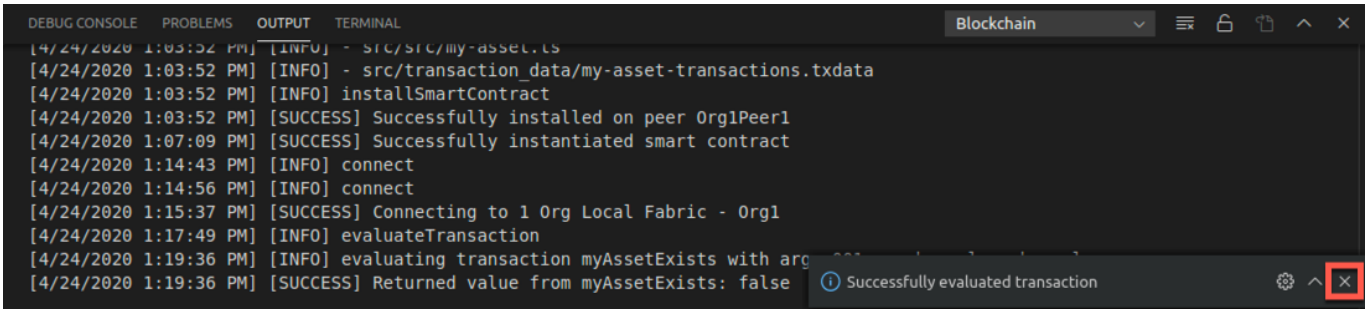
- `A4.8`: Review the output



When the method completes, the Transaction output field will show the results of the evaluation, and as expected the result is false.

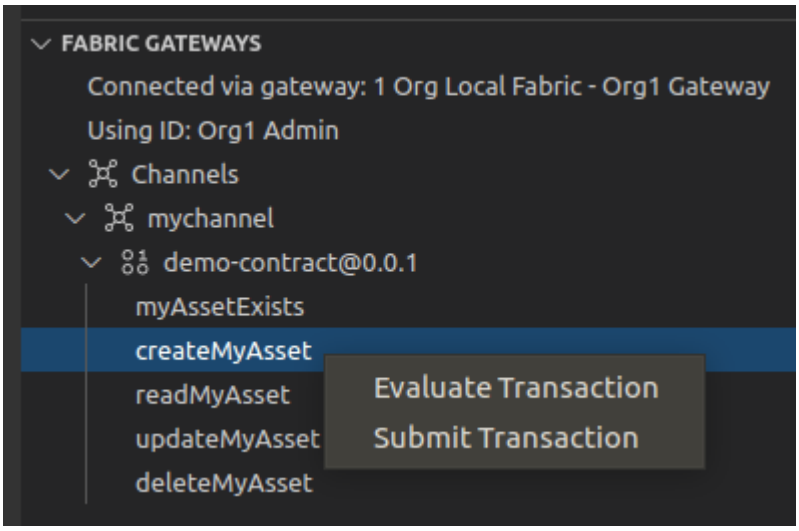
The result of the transaction will also be shown in the output view at the bottom of the screen.

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



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`: On the Transaction view click on the dropdown for Transaction Name and select createMyAsset.



As you may recall, the createMyAsset transaction takes two arguments: a key and its associated value.

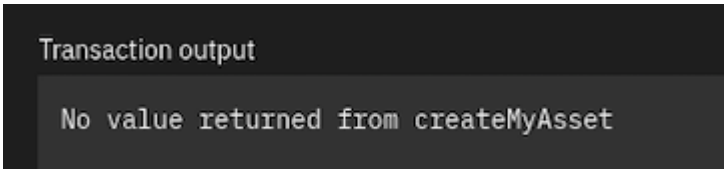
A4.11: Replace the Transaction arguments with

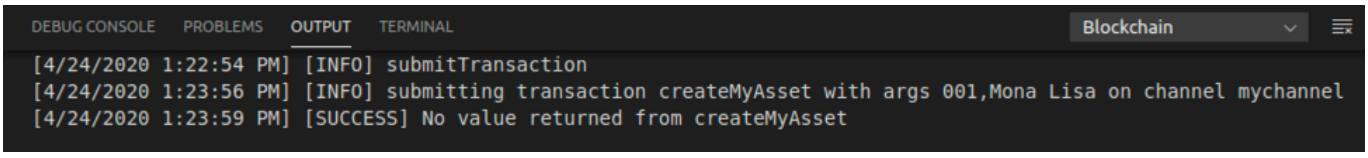
```
{
  "myAssetId": "001",
  "value": "Mona Lisa"
}
```

Take care with the syntax of the JSON argument pairs.

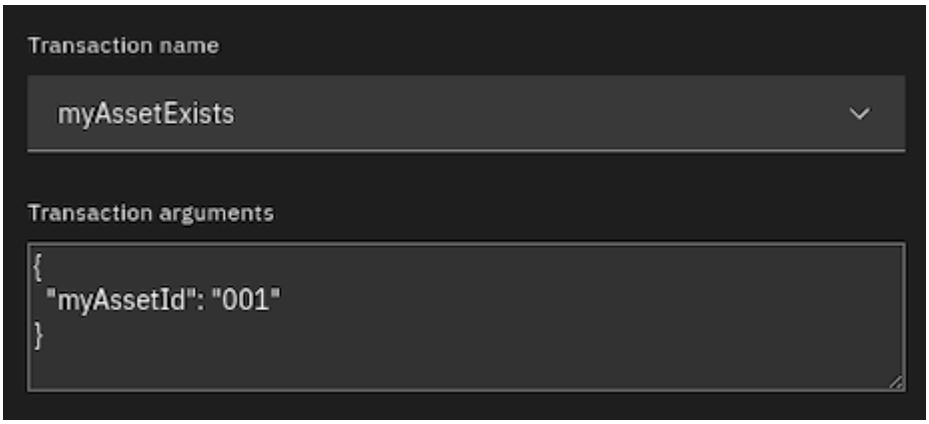
A4.12: Check the Transaction name and the Transaction arguments, then click Submit transaction.

The Transaction output on the Transaction View will show that no value was returned, and the output log at the bottom of the screen shows that the transaction was successful.

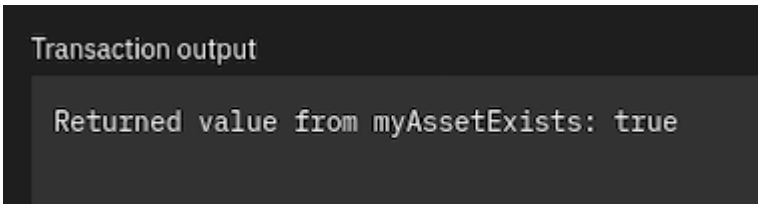




A4.13: Still on the Transaction view, select "myAssetExists" as the Transaction name, specify `"001"` for myAssetId in the JSON arguments and click Evaluate transaction.



The returned value seen in the Transaction output is now 'true'.



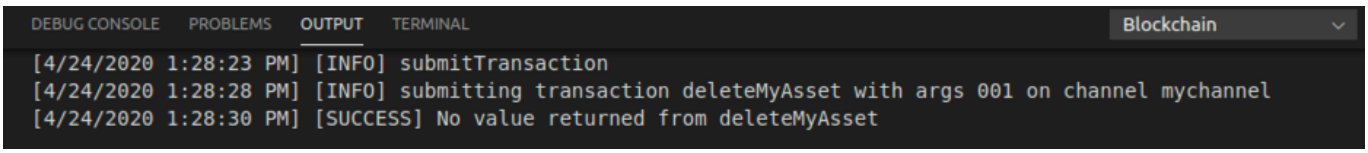
A4.14: Submit the "updateMyAsset" transaction to change the value of the "001" key to "The Hay Wain". The JSON Transaction arguments will be:

```
{
  "myAssetId": "001",
  "value": "The Hay Wain"
}
```

No value will be returned from the transaction.

A4.15: Evaluate the "readMyAsset" transaction to return the updated asset. Specify `"001"` for myAssetId.

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



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.

→ **Next: A5: Invoking a smart contract from an external application**