peer lifecycle chaincode

The `peer lifecycle chaincode` subcommand allows administrators to use the Fabric chaincode lifecycle to package a chaincode, install it on your peers, approve a chaincode definition for your organization, and then commit the definition to a channel. The chaincode is ready to be used after the definition has been successfully committed to the channel. For more information, visit Fabric chaincode lifecycle.

Note: These instructions use the Fabric chaincode lifecycle introduced in the v2.0 release. If you would like to use the old lifecycle to install and instantiate a chaincode, visit the peer chaincode command reference.

Syntax

The `peer lifecycle chaincode` command has the following subcommands:

- package
- install
- queryinstalled
- getinstalledpackage
- approveformyorg
- queryapproved
- checkcommitreadiness
- commit
- querycommitted

Each peer lifecycle chaincode subcommand is described together with its options in its own section in this topic.

peer lifecycle

```
Perform _lifecycle operations

Usage:
  peer lifecycle [command]

Available Commands:
  chaincode   Perform chaincode operations: package|install|queryinstalled|getinstalledpackage|ap

Flags:
  -h, --help   help for lifecycle

Use "peer lifecycle [command] --help" for more information about a command.
```

peer lifecycle chaincode

```
Perform chaincode operations: package|install|queryinstalled|getinstalledpackage|approveformyorg|

Usage:
  peer lifecycle chaincode [command]
```

```
Available Commands:
  approveformyorg      Approve the chaincode definition for my org.
  checkcommitreadiness Check whether a chaincode definition is ready to be committed on a channel
  commit               Commit the chaincode definition on the channel.
  getinstalledpackage  Get an installed chaincode package from a peer.
  install              Install a chaincode.
  package              Package a chaincode
  queryapproved        Query an org's approved chaincode definition from its peer.
  querycommitted       Query the committed chaincode definitions by channel on a peer.
  queryinstalled       Query the installed chaincodes on a peer.

Flags:
      --cafile string                       Path to file containing PEM-encoded trusted certifica
      --certfile string                     Path to file containing PEM-encoded X509 public key t
      --clientauth                          Use mutual TLS when communicating with the orderer en
      --connTimeout duration                Timeout for client to connect (default 3s)
  -h, --help                                help for chaincode
      --keyfile string                      Path to file containing PEM-encoded private key to us
  -o, --orderer string                      Ordering service endpoint
      --ordererTLSHostnameOverride string   The hostname override to use when validating the TLS
      --tls                                 Use TLS when communicating with the orderer endpoint

Use "peer lifecycle chaincode [command] --help" for more information about a command.
```

peer lifecycle chaincode package

```
Package a chaincode and write the package to a file.

Usage:
  peer lifecycle chaincode package [outputfile] [flags]

Flags:
      --connectionProfile string       The fully qualified path to the connection profile that pr
  -h, --help                           help for package
      --label string                   The package label contains a human-readable description of
  -l, --lang string                    Language the chaincode is written in (default "golang")
  -p, --path string                    Path to the chaincode
      --peerAddresses stringArray      The addresses of the peers to connect to
      --tlsRootCertFiles stringArray   If TLS is enabled, the paths to the TLS root cert files of

Global Flags:
      --cafile string                       Path to file containing PEM-encoded trusted certifica
      --certfile string                     Path to file containing PEM-encoded X509 public key t
      --clientauth                          Use mutual TLS when communicating with the orderer en
      --connTimeout duration                Timeout for client to connect (default 3s)
      --keyfile string                      Path to file containing PEM-encoded private key to us
  -o, --orderer string                      Ordering service endpoint
      --ordererTLSHostnameOverride string   The hostname override to use when validating the TLS
      --tls                                 Use TLS when communicating with the orderer endpoint
```

peer lifecycle chaincode install

```
Install a chaincode on a peer.

Usage:
  peer lifecycle chaincode install [flags]

Flags:
      --connectionProfile string       The fully qualified path to the connection profile that pr
  -h, --help                           help for install
      --peerAddresses stringArray      The addresses of the peers to connect to
      --targetPeer string              When using a connection profile, the name of the peer to t
      --tlsRootCertFiles stringArray   If TLS is enabled, the paths to the TLS root cert files of

Global Flags:
      --cafile string                       Path to file containing PEM-encoded trusted certifica
      --certfile string                     Path to file containing PEM-encoded X509 public key t
      --clientauth                          Use mutual TLS when communicating with the orderer en
      --connTimeout duration                Timeout for client to connect (default 3s)
      --keyfile string                      Path to file containing PEM-encoded private key to us
```

```
  -o, --orderer string                       Ordering service endpoint
      --ordererTLSHostnameOverride string    The hostname override to use when validating the TLS
      --tls                                  Use TLS when communicating with the orderer endpoint
```

peer lifecycle chaincode queryinstalled

```
Query the installed chaincodes on a peer.

Usage:
  peer lifecycle chaincode queryinstalled [flags]

Flags:
      --connectionProfile string       The fully qualified path to the connection profile that pr
  -h, --help                           help for queryinstalled
  -O, --output string                  The output format for query results. Default is human-read
      --peerAddresses stringArray      The addresses of the peers to connect to
      --targetPeer string              When using a connection profile, the name of the peer to t
      --tlsRootCertFiles stringArray   If TLS is enabled, the paths to the TLS root cert files of

Global Flags:
      --cafile string                        Path to file containing PEM-encoded trusted certifica
      --certfile string                      Path to file containing PEM-encoded X509 public key t
      --clientauth                           Use mutual TLS when communicating with the orderer en
      --connTimeout duration                 Timeout for client to connect (default 3s)
      --keyfile string                       Path to file containing PEM-encoded private key to us
  -o, --orderer string                       Ordering service endpoint
      --ordererTLSHostnameOverride string    The hostname override to use when validating the TLS
      --tls                                  Use TLS when communicating with the orderer endpoint
```

peer lifecycle chaincode getinstalledpackage

```
Get an installed chaincode package from a peer.

Usage:
  peer lifecycle chaincode getinstalledpackage [outputfile] [flags]

Flags:
      --connectionProfile string       The fully qualified path to the connection profile that pr
  -h, --help                           help for getinstalledpackage
      --output-directory string        The output directory to use when writing a chaincode insta
      --package-id string              The identifier of the chaincode install package
      --peerAddresses stringArray      The addresses of the peers to connect to
      --targetPeer string              When using a connection profile, the name of the peer to t
      --tlsRootCertFiles stringArray   If TLS is enabled, the paths to the TLS root cert files of

Global Flags:
      --cafile string                        Path to file containing PEM-encoded trusted certifica
      --certfile string                      Path to file containing PEM-encoded X509 public key t
      --clientauth                           Use mutual TLS when communicating with the orderer en
      --connTimeout duration                 Timeout for client to connect (default 3s)
      --keyfile string                       Path to file containing PEM-encoded private key to us
  -o, --orderer string                       Ordering service endpoint
      --ordererTLSHostnameOverride string    The hostname override to use when validating the TLS
      --tls                                  Use TLS when communicating with the orderer endpoint
```

peer lifecycle chaincode approveformyorg

```
Approve the chaincode definition for my organization.

Usage:
  peer lifecycle chaincode approveformyorg [flags]

Flags:
      --channel-config-policy string   The endorsement policy associated to this chaincode specif
  -C, --channelID string               The channel on which this command should be executed
      --collections-config string      The fully qualified path to the collection JSON file inclu
      --connectionProfile string       The fully qualified path to the connection profile that pr
```

```
  -E, --endorsement-plugin string      The name of the endorsement plugin to be used for this cha
  -h, --help                           help for approveformyorg
      --init-required                  Whether the chaincode requires invoking 'init'
  -n, --name string                    Name of the chaincode
      --package-id string              The identifier of the chaincode install package
      --peerAddresses stringArray      The addresses of the peers to connect to
      --sequence int                   The sequence number of the chaincode definition for the ch
      --signature-policy string        The endorsement policy associated to this chaincode specif
      --tlsRootCertFiles stringArray   If TLS is enabled, the paths to the TLS root cert files of
  -V, --validation-plugin string       The name of the validation plugin to be used for this chai
  -v, --version string                 Version of the chaincode
      --waitForEvent                   Whether to wait for the event from each peer's deliver fil
      --waitForEventTimeout duration   Time to wait for the event from each peer's deliver filter

Global Flags:
      --cafile string                       Path to file containing PEM-encoded trusted certifica
      --certfile string                     Path to file containing PEM-encoded X509 public key t
      --clientauth                          Use mutual TLS when communicating with the orderer en
      --connTimeout duration                Timeout for client to connect (default 3s)
      --keyfile string                      Path to file containing PEM-encoded private key to us
  -o, --orderer string                      Ordering service endpoint
      --ordererTLSHostnameOverride string   The hostname override to use when validating the TLS
      --tls                                 Use TLS when communicating with the orderer endpoint
```

peer lifecycle chaincode queryapproved

```
Query an organization's approved chaincode definition from its peer.

Usage:
  peer lifecycle chaincode queryapproved [flags]

Flags:
  -C, --channelID string               The channel on which this command should be executed
      --connectionProfile string       The fully qualified path to the connection profile that pr
  -h, --help                           help for queryapproved
  -n, --name string                    Name of the chaincode
  -O, --output string                  The output format for query results. Default is human-read
      --peerAddresses stringArray      The addresses of the peers to connect to
      --sequence int                   The sequence number of the chaincode definition for the ch
      --tlsRootCertFiles stringArray   If TLS is enabled, the paths to the TLS root cert files of

Global Flags:
      --cafile string                       Path to file containing PEM-encoded trusted certifica
      --certfile string                     Path to file containing PEM-encoded X509 public key t
      --clientauth                          Use mutual TLS when communicating with the orderer en
      --connTimeout duration                Timeout for client to connect (default 3s)
      --keyfile string                      Path to file containing PEM-encoded private key to us
  -o, --orderer string                      Ordering service endpoint
      --ordererTLSHostnameOverride string   The hostname override to use when validating the TLS
      --tls                                 Use TLS when communicating with the orderer endpoint
```

peer lifecycle chaincode checkcommitreadiness

```
Check whether a chaincode definition is ready to be committed on a channel.

Usage:
  peer lifecycle chaincode checkcommitreadiness [flags]

Flags:
      --channel-config-policy string   The endorsement policy associated to this chaincode specif
  -C, --channelID string               The channel on which this command should be executed
      --collections-config string      The fully qualified path to the collection JSON file inclu
      --connectionProfile string       The fully qualified path to the connection profile that pr
  -E, --endorsement-plugin string      The name of the endorsement plugin to be used for this cha
  -h, --help                           help for checkcommitreadiness
      --init-required                  Whether the chaincode requires invoking 'init'
  -n, --name string                    Name of the chaincode
  -O, --output string                  The output format for query results. Default is human-read
      --peerAddresses stringArray      The addresses of the peers to connect to
      --sequence int                   The sequence number of the chaincode definition for the ch
      --signature-policy string        The endorsement policy associated to this chaincode specif
```

```
      --tlsRootCertFiles stringArray   If TLS is enabled, the paths to the TLS root cert files of
  -V, --validation-plugin string       The name of the validation plugin to be used for this chai
  -v, --version string                 Version of the chaincode

Global Flags:
      --cafile string                          Path to file containing PEM-encoded trusted certifica
      --certfile string                        Path to file containing PEM-encoded X509 public key t
      --clientauth                             Use mutual TLS when communicating with the orderer en
      --connTimeout duration                   Timeout for client to connect (default 3s)
      --keyfile string                         Path to file containing PEM-encoded private key to us
  -o, --orderer string                         Ordering service endpoint
      --ordererTLSHostnameOverride string      The hostname override to use when validating the TLS
      --tls                                    Use TLS when communicating with the orderer endpoint
```

peer lifecycle chaincode commit

```
Commit the chaincode definition on the channel.

Usage:
  peer lifecycle chaincode commit [flags]

Flags:
      --channel-config-policy string   The endorsement policy associated to this chaincode specif
  -C, --channelID string               The channel on which this command should be executed
      --collections-config string      The fully qualified path to the collection JSON file inclu
      --connectionProfile string       The fully qualified path to the connection profile that pr
  -E, --endorsement-plugin string      The name of the endorsement plugin to be used for this cha
  -h, --help                           help for commit
      --init-required                  Whether the chaincode requires invoking 'init'
  -n, --name string                    Name of the chaincode
      --peerAddresses stringArray      The addresses of the peers to connect to
      --sequence int                   The sequence number of the chaincode definition for the ch
      --signature-policy string        The endorsement policy associated to this chaincode specif
      --tlsRootCertFiles stringArray   If TLS is enabled, the paths to the TLS root cert files of
  -V, --validation-plugin string       The name of the validation plugin to be used for this chai
  -v, --version string                 Version of the chaincode
      --waitForEvent                   Whether to wait for the event from each peer's deliver fil
      --waitForEventTimeout duration   Time to wait for the event from each peer's deliver filter

Global Flags:
      --cafile string                          Path to file containing PEM-encoded trusted certifica
      --certfile string                        Path to file containing PEM-encoded X509 public key t
      --clientauth                             Use mutual TLS when communicating with the orderer en
      --connTimeout duration                   Timeout for client to connect (default 3s)
      --keyfile string                         Path to file containing PEM-encoded private key to us
  -o, --orderer string                         Ordering service endpoint
      --ordererTLSHostnameOverride string      The hostname override to use when validating the TLS
      --tls                                    Use TLS when communicating with the orderer endpoint
```

peer lifecycle chaincode querycommitted

```
Query the committed chaincode definitions by channel on a peer. Optional: provide a chaincode nam

Usage:
  peer lifecycle chaincode querycommitted [flags]

Flags:
  -C, --channelID string               The channel on which this command should be executed
      --connectionProfile string       The fully qualified path to the connection profile that pr
  -h, --help                           help for querycommitted
  -n, --name string                    Name of the chaincode
  -O, --output string                  The output format for query results. Default is human-read
      --peerAddresses stringArray      The addresses of the peers to connect to
      --tlsRootCertFiles stringArray   If TLS is enabled, the paths to the TLS root cert files of

Global Flags:
      --cafile string                          Path to file containing PEM-encoded trusted certifica
      --certfile string                        Path to file containing PEM-encoded X509 public key t
      --clientauth                             Use mutual TLS when communicating with the orderer en
      --connTimeout duration                   Timeout for client to connect (default 3s)
      --keyfile string                         Path to file containing PEM-encoded private key to us
```

```
  -o, --orderer string                       Ordering service endpoint
      --ordererTLSHostnameOverride string    The hostname override to use when validating the TLS
      --tls                                  Use TLS when communicating with the orderer endpoint
```

Example Usage

peer lifecycle chaincode package example

A chaincode needs to be packaged before it can be installed on your peers. This example uses the `peer lifecycle chaincode package` command to package a Go chaincode.

- Use the `--path` flag to indicate the location of the chaincode. The path must be a fully qualified path or a path relative to your present working directory.
- Use the `--label` flag to provide a chaincode package label of `myccv1` that your organization will use to identify the package.

```
peer lifecycle chaincode package mycc.tar.gz --path $CHAINCODE_DIR --lang golang --label myccv
```

peer lifecycle chaincode install example

After the chaincode is packaged, you can use the `peer chaincode install` command to install the chaincode on your peers.

- Install the `mycc.tar.gz` package on `peer0.org1.example.com:7051` (the peer defined by `--peerAddresses`).

```
peer lifecycle chaincode install mycc.tar.gz --peerAddresses peer0.org1.example.com:7051
```

If successful, the command will return the package identifier. The package ID is the package label combined with a hash of the chaincode package taken by the peer.

```
2019-03-13 13:48:53.691 UTC [cli.lifecycle.chaincode] submitInstallProposal -> INFO 001 Instal
2019-03-13 13:48:53.691 UTC [cli.lifecycle.chaincode] submitInstallProposal -> INFO 002 Chainc
```

peer lifecycle chaincode queryinstalled example

You need to use the chaincode package identifier to approve a chaincode definition for your organization. You can find the package ID for the chaincodes you have installed by using the `peer lifecycle chaincode queryinstalled` command:

```
peer lifecycle chaincode queryinstalled --peerAddresses peer0.org1.example.com:7051
```

A successful command will return the package ID associated with the package label.

```
Get installed chaincodes on peer:
Package ID: myccv1:a7ca45a7cc85f1d89c905b775920361ed089a364e12a9b6d55ba75c965ddd6a9, Label: myccv
```

- You can also use the `--output` flag to have the CLI format the output as JSON.

```
peer lifecycle chaincode queryinstalled --peerAddresses peer0.org1.example.com:7051 --output j
```

If successful, the command will return the chaincodes you have installed as JSON.

```
{
  "installed_chaincodes": [
    {
      "package_id": "mycc_1:aab9981fa5649cfe25369fce7bb5086a69672a631e4f95c4af1b5198fe9f845b",
      "label": "mycc_1",
      "references": {
        "mychannel": {
          "chaincodes": [
            {
              "name": "mycc",
              "version": "1"
            }
          ]
        }
      }
    }
  ]
}
```

peer lifecycle chaincode getinstalledpackage example

You can retrieve an installed chaincode package from a peer using the `peer lifecycle chaincode getinstalledpackage` command. Use the package identifier returned by `queryinstalled`.

- Use the `--package-id` flag to pass in the chaincode package identifier. Use the `--output-directory` flag to specify where to write the chaincode package. If the output directory is not specified, the chaincode package will be written in the current directory.

```
peer lifecycle chaincode getinstalledpackage --package-id myccv1:a7ca45a7cc85f1d89c905b775920361e
```

peer lifecycle chaincode approveformyorg example

Once the chaincode package has been installed on your peers, you can approve a chaincode definition for your organization. The chaincode definition includes the important parameters of chaincode governance, including the chaincode name, version and the endorsement policy.

Here is an example of the `peer lifecycle chaincode approveformyorg` command, which approves the definition of a chaincode named `mycc` at version `1.0` on channel `mychannel`.

- Use the `--package-id` flag to pass in the chaincode package identifier. Use the `--signature-policy` flag to define an endorsement policy for the chaincode. Use the `init-required` flag to request the execution of the `Init` function to initialize the chaincode.

```
export ORDERER_CA=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizatio

peer lifecycle chaincode approveformyorg  -o orderer.example.com:7050 --tls --cafile $ORDERER_

2019-03-18 16:04:09.046 UTC [cli.lifecycle.chaincode] InitCmdFactory -> INFO 001 Retrieved cha
2019-03-18 16:04:11.253 UTC [chaincodeCmd] ClientWait -> INFO 002 txid [efba188ca77889cc1c328f
```

- You can also use the `--channel-config-policy` flag use a policy inside the channel configuration as the chaincode endorsement policy. The default endorsement policy is `Channel/Application/Endorsement`

```
export ORDERER_CA=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizatio

peer lifecycle chaincode approveformyorg -o orderer.example.com:7050 --tls --cafile $ORDERER_C

2019-03-18 16:04:09.046 UTC [cli.lifecycle.chaincode] InitCmdFactory -> INFO 001 Retrieved cha
2019-03-18 16:04:11.253 UTC [chaincodeCmd] ClientWait -> INFO 002 txid [efba188ca77889cc1c328f
```

peer lifecycle chaincode queryapproved example

You can query an organization's approved chaincode definition by using the `peer lifecycle chaincode queryapproved` command. You can use this command to see the details (including package ID) of approved chaincode definitions.

- Here is an example of the `peer lifecycle chaincode queryapproved` command, which queries the approved definition of a chaincode named `mycc` at sequence number `1` on channel `mychannel`.

```
peer lifecycle chaincode queryapproved -C mychannel -n mycc --sequence 1

Approved chaincode definition for chaincode 'mycc' on channel 'mychannel':
sequence: 1, version: 1, init-required: true, package-id: mycc_1:d02f72000e7c0f715840f51cb8d72
```

  If NO package is specified for the approved definition, this command will display an empty package ID.
- You can also use this command without specifying the sequence number in order to query the latest approved definition (latest: the newer of the currently defined sequence number and the next sequence number).

```
peer lifecycle chaincode queryapproved -C mychannel -n mycc

Approved chaincode definition for chaincode 'mycc' on channel 'mychannel':
sequence: 3, version: 3, init-required: false, package-id: mycc_1:d02f72000e7c0f715840f51cb8d7
```

- You can also use the `--output` flag to have the CLI format the output as JSON.
  - When querying an approved chaincode definition for which package is specified

    ```
    peer lifecycle chaincode queryapproved -C mychannel -n mycc --sequence 1 --output json
    ```

    If successful, the command will return a JSON that has the approved chaincode definition for chaincode `mycc` at sequence number `1` on channel `mychannel`.

```
{
  "sequence": 1,
  "version": "1",
  "endorsement_plugin": "escc",
  "validation_plugin": "vscc",
  "validation_parameter": "EiAvQ2hhbm5lbC9BcHBsaWNhdGlvbi9FbmRvcnNlbWVudA==",
  "collections": {},
  "init_required": true,
  "source": {
    "Type": {
      "LocalPackage": {
        "package_id": "mycc_1:d02f72000e7c0f715840f51cb8d72d70bc1ba230552f8445dded0ec8b6e0b
      }
    }
  }
}
```


- When querying an approved chaincode definition for which package is NOT specified

```
peer lifecycle chaincode queryapproved -C mychannel -n mycc --sequence 2 --output json
```

If successful, the command will return a JSON that has the approved chaincode definition for chaincode `mycc` at sequence number `2` on channel `mychannel`.


```
{
  "sequence": 2,
  "version": "2",
  "endorsement_plugin": "escc",
  "validation_plugin": "vscc",
  "validation_parameter": "EiAvQ2hhbm5lbC9BcHBsaWNhdGlvbi9FbmRvcnNlbWVudA==",
  "collections": {},
  "source": {
    "Type": {
      "Unavailable": {}
    }
  }
}
```


peer lifecycle chaincode checkcommitreadiness example

You can check whether a chaincode definition is ready to be committed using the `peer lifecycle chaincode checkcommitreadiness` command, which will return successfully if a subsequent commit of the definition is expected to succeed. It also outputs which organizations have approved the chaincode definition. If an organization has approved the chaincode definition specified in the command, the command will return a value of true. You can use this command to learn whether enough channel members have approved a chaincode definition to meet the `Application/Channel/Endorsement` policy (a majority by default) before the definition can be committed to a channel.

- Here is an example of the `peer lifecycle chaincode checkcommitreadiness` command, which checks a chaincode named `mycc` at version `1.0` on channel `mychannel`.

```
export ORDERER_CA=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizatio

peer lifecycle chaincode checkcommitreadiness -o orderer.example.com:7050 --channelID mychanne
```

If successful, the command will return the organizations that have approved the chaincode definition.

```
Chaincode definition for chaincode 'mycc', version '1.0', sequence '1' on channel
'mychannel' approval status by org:
Org1MSP: true
Org2MSP: true
```

- You can also use the `--output` flag to have the CLI format the output as JSON.

```
export ORDERER_CA=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizatio

peer lifecycle chaincode checkcommitreadiness -o orderer.example.com:7050 --channelID mychanne
```

If successful, the command will return a JSON map that shows if an organization has approved the chaincode definition.

```
{
   "Approvals": {
      "Org1MSP": true,
      "Org2MSP": true
   }
}
```

peer lifecycle chaincode commit example

Once a sufficient number of organizations approve a chaincode definition for their organizations (a majority by default), one organization can commit the definition the channel using the `peer lifecycle chaincode commit` command:

- This command needs to target the peers of other organizations on the channel to collect their organization endorsement for the definition.

```
export ORDERER_CA=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizatio

peer lifecycle chaincode commit -o orderer.example.com:7050 --channelID mychannel --name mycc

2019-03-18 16:14:27.258 UTC [chaincodeCmd] ClientWait -> INFO 001 txid [b6f657a14689b27d69a50f
2019-03-18 16:14:27.321 UTC [chaincodeCmd] ClientWait -> INFO 002 txid [b6f657a14689b27d69a50f
```

peer lifecycle chaincode querycommitted example

You can query the chaincode definitions that have been committed to a channel by using the `peer lifecycle chaincode querycommitted` command. You can use this command to query the current definition sequence number before upgrading a chaincode.

- You need to supply the chaincode name and channel name in order to query a specific chaincode definition and the organizations that have approved it.

```
export ORDERER_CA=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizatio

peer lifecycle chaincode querycommitted -o orderer.example.com:7050 --channelID mychannel --na

Committed chaincode definition for chaincode 'mycc' on channel 'mychannel':
```

```
Version: 1, Sequence: 1, Endorsement Plugin: escc, Validation Plugin: vscc
Approvals: [Org1MSP: true, Org2MSP: true]
```

- You can also specify just the channel name in order to query all chaincode definitions on that channel.

```
export ORDERER_CA=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizatio

peer lifecycle chaincode querycommitted -o orderer.example.com:7050 --channelID mychannel --tl

Committed chaincode definitions on channel 'mychannel':
Name: mycc, Version: 1, Sequence: 1, Endorsement Plugin: escc, Validation Plugin: vscc
Name: yourcc, Version: 2, Sequence: 3, Endorsement Plugin: escc, Validation Plugin: vscc
```

- You can also use the `--output` flag to have the CLI format the output as JSON.
  - For querying a specific chaincode definition

```
export ORDERER_CA=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganiza

peer lifecycle chaincode querycommitted -o orderer.example.com:7050 --channelID mychannel -
```

    If successful, the command will return a JSON that has committed chaincode definition for chaincode ‘mycc’ on channel ‘mychannel’.

```
{
  "sequence": 1,
  "version": "1",
  "endorsement_plugin": "escc",
  "validation_plugin": "vscc",
  "validation_parameter": "EiAvQ2hhbm5lbC9BcHBsaWNhdGlvbi9FbmRvcnNlbWVudA==",
  "collections": {},
  "init_required": true,
  "approvals": {
    "Org1MSP": true,
    "Org2MSP": true
  }
}
```

    The `validation_parameter` is base64 encoded. An example of the command to decode it is as follows.

```
echo EiAvQ2hhbm5lbC9BcHBsaWNhdGlvbi9FbmRvcnNlbWVudA== | base64 -d

 /Channel/Application/Endorsement
```

  - For querying all chaincode definitions on that channel

```
export ORDERER_CA=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganiza

peer lifecycle chaincode querycommitted -o orderer.example.com:7050 --channelID mychannel -
```

    If successful, the command will return a JSON that has committed chaincode definitions on channel ‘mychannel’.

```
{
  "chaincode_definitions": [
```

```
    {
      "name": "mycc",
      "sequence": 1,
      "version": "1",
      "endorsement_plugin": "escc",
      "validation_plugin": "vscc",
      "validation_parameter": "EiAvQ2hhbm5lbC9BcHBsaWNhdGlvbi9FbmRvcnNlbWVudA==",
      "collections": {},
      "init_required": true
    },
    {
      "name": "yourcc",
      "sequence": 3,
      "version": "2",
      "endorsement_plugin": "escc",
      "validation_plugin": "vscc",
      "validation_parameter": "EiAvQ2hhbm5lbC9BcHBsaWNhdGlvbi9FbmRvcnNlbWVudA==",
      "collections": {}
    }
  ]
}
```