启用新的链码生命周期

Enabling the new chaincode lifecycle

从1.4升级到2.0的用户将需要修改通道配置以启用新的生命周期特性。这个过程涉及到一系列相关用户必须要执行的通道配置更新。

Users upgrading from v1.4.x to v2.x will have to edit their channel configurations to enable the new lifecycle features. This process involves a series of channel configuration updates the relevant users will have to perform.

需要注意的是，你的应用通道的 `Channel` 和 `Application` 能力（capabilities） 需要更新到 `V2_0`，以确保新的链码生命周期正常工作。 更多详情请参考 获取2.0的注意事项。

Note that the `Channel` and `Application` capabilities of your application channels will have to be updated to `V2_0` for the new chaincode lifecycle to work. Check out Considerations for getting to 2.0 for more information.

整体来看，每一个通道配置升级需要执行三个步骤：

Updating a channel configuration is, at a high level, a three step process (for each channel):

1. 获取最新的通道配置
2. 创建一个修改后的通道配置
3. 创建一个配置更新交易
4. Get the latest channel config
5. Create a modified channel config
6. Create a config update transaction

我们将借助文件 `enable_lifecycle.json` 来执行这些通道配置更新, 这个文件包含了我们需要做的所有通道配置更新。需要注意的是，在生产配置中很可能多个用户都会发出这些通道更新请求。但是，为了简单起见，我们将在一个文件中表示所有用户的更新。

We will be performing these channel configuration updates by leveraging a file called `enable_lifecycle.json`, which contains all of the updates we will be making in the channel configurations. Note that in a production setting it is likely that multiple users would be making these channel update requests. However, for the sake of simplicity, we are presenting all of the updates as how they would appear in a single file.

创建 `enable_lifecycle.json`

Create `enable_lifecycle.json`

注意，除了 `enable_lifecycle.json`，还需要使用 `jq` 对修改后的配置文件进行编辑。修改后的配置文件也可以手动修改（在文件被拉取、翻译和检查后）。详情请参考通道配置示例。

Note that in addition to using `enable_lifecycle.json`, this tutorial also uses `jq` to apply the edits to the modified config file. The modified config can also be edited manually (after it has been pulled, translated, and scoped). Check out this sample channel configuration for reference.

但是，这里描述的过程（使用一个 JSON 文件和一个像 `jq` 这样的工具）确实具备可编写脚本的优势，适合提交大批量的通道配置更新，也是编辑一个通道配置的推荐过程。

However, the process described here (using a JSON file and a tool like `jq`) does have the advantage of being scriptable, making it suitable for proposing configuration updates to a large number of channels, and is the recommended process for editing a channel configuration.

需要注意的是 `enable_lifecycle.json` 使用的是示例值，比如说 `org1Policies` 和 `Org1ExampleCom`，当你部署时需将值特殊处理：

Note that the `enable_lifecycle.json` uses sample values, for example `org1Policies` and the `Org1ExampleCom`, which will be specific to your deployment):

```
{
  "org1Policies": {
      "Endorsement": {
           "mod_policy": "Admins",
           "policy": {
               "type": 1,
               "value": {
               "identities": [
                  {
                     "principal": {
                         "msp_identifier": "Org1ExampleCom",
                         "role": "PEER"
                     },
                     "principal_classification": "ROLE"
                  }
                ],
                "rule": {
                  "n_out_of": {
                       "n": 1,
                       "rules": [
                        {
                            "signed_by": 0
                        }
                       ]
                   }
                },
                "version": 0
              }
           },
           "version": "0"
      }
   },
  "org2Policies": {
      "Endorsement": {
           "mod_policy": "Admins",
           "policy": {
               "type": 1,
               "value": {
               "identities": [
                  {
                     "principal": {
                         "msp_identifier": "Org2ExampleCom",
                         "role": "PEER"
```

```
                    },
                    "principal_classification": "ROLE"
                }
              ],
              "rule": {
                "n_out_of": {
                    "n": 1,
                    "rules": [
                     {
                         "signed_by": 0
                     }
                    ]
                }
              },
              "version": 0
            }
        },
        "version": "0"
    }
  },
  "appPolicies": {
      "Endorsement": {
          "mod_policy": "Admins",
          "policy": {
              "type": 3,
              "value": {
                  "rule": "MAJORITY",
                  "sub_policy": "Endorsement"
              }
          },
          "version": "0"
      },
      "LifecycleEndorsement": {
          "mod_policy": "Admins",
          "policy": {
              "type": 3,
              "value": {
                  "rule": "MAJORITY",
                  "sub_policy": "Endorsement"
              }
          },
          "version": "0"
      }
  },
  "acls": {
      "_lifecycle/CheckCommitReadiness": {
          "policy_ref": "/Channel/Application/Writers"
      },
      "_lifecycle/CommitChaincodeDefinition": {
          "policy_ref": "/Channel/Application/Writers"
      },
      "_lifecycle/QueryChaincodeDefinition": {
          "policy_ref": "/Channel/Application/Readers"
      },
      "_lifecycle/QueryChaincodeDefinitions": {
          "policy_ref": "/Channel/Application/Readers"
      }
  }
}
```

注意:如果这个组织启用了 NodeOUs 并且不属于 `'MEMBER'` 的话，这些新策略中的 “role” 字段应该设置成 `'PEER'` 。

Note: the “role” field of these new policies should say `'PEER'` if NodeOUs are enabled for the org, and `'MEMBER'` if they are not.

编辑通道配置

Edit the channel configurations

系统通道更新

System channel updates

因为使用新生命周期的系统通道配置修改，仅会涉及到系统通道配置内的节点组织配置参数，所以被编辑的每个节点组织都需要对相关的通道配置进行签名更新。

Because configuration changes to the system channel to enable the new lifecycle only involve parameters inside the configuration of the peer organizations within the channel configuration, each peer organization being edited will have to sign the relevant channel configuration update.

然而, 在默认情况下, 系统通道只能被系统管理员编辑（通常这些人是排序服务组织的管理员，而不是节点组织的管理员）, 也就是说联盟中的节点组织的配置更新只能由系统通道的管理员提议和发送到相关节点组织进行签名。

However, by default, the system channel can only be edited by system channel admins (typically these are admins of the ordering service organizations and not peer organizations), which means that the configuration updates to the peer organizations in the consortium will have to be proposed by a system channel admin and sent to the relevant peer organization to be signed.

你需要导出以下变量:

You will need to export the following variables:

- `CH_NAME`: 将要被更新的系统通道名称。
- `CORE_PEER_LOCALMSPID`: 提交的通道更新组织的 MSP ID。这是排序服务组织中一员的MSP。
- `CORE_PEER_MSPCONFIGPATH`: 代表组织 MSP 的绝对路径。
- `TLS_ROOT_CA`: 提交的系统通道更新的组织的根 CA 证书的绝对路径。
- `ORDERER_CONTAINER`: 一个排序节点容器的名称。当定位排序服务时，你可以在排序服务中定位任意一个特定节点。而你的请求会被自动推送至领导者节点。
- `ORGNAME`: 你正在更新的组织名称。
- `CONSORTIUM_NAME`: 被更新联盟的名称。

环境变量设置完毕后，导航至 步骤 1：拉取和翻译配置。

- `CH_NAME`: the name of the system channel being updated.
- `CORE_PEER_LOCALMSPID`: the MSP ID of the organization proposing the channel update. This will be the MSP of one of the ordering service organizations.
- `CORE_PEER_MSPCONFIGPATH`: the absolute path to the MSP representing your organization.
- `TLS_ROOT_CA`: the absolute path to the root CA certificate of the organization proposing the system channel update.
- `ORDERER_CONTAINER`: the name of an ordering node container. When targeting the ordering service, you can target any particular node in the ordering service. Your requests will be forwarded to the leader automatically.
- `ORGNAME`: the name of the organization you are currently updating.

- `CONSORTIUM_NAME`: the name of the consortium being updated.

`modified_config.json` 设置完毕后，使用以下命令添加生命周期组织策略（如 `enable_lifecycle.json` 文件中所展示的）：

```
jq -s ".[0] * {\"channel_group\":{\"groups\":{\"Consortiums\":{\"groups\": {\"$CONSORTIUM_NAME\":
```

Once you have set the environment variables, navigate to Step 1: Pull and translate the config.

紧接着, 按照步骤[步骤 3：重编码和提交配置](./config_update.html#step-3-re-encode-and-submit-the-config)。

Then, add the lifecycle organization policy (as listed in `enable_lifecycle.json`) to a file called `modified_config.json` using this command:

综上所述, 这些变更需要由系统通道管理员提议并发送到相关节点组织进行签名。

```
jq -s ".[0] * {\"channel_group\":{\"groups\":{\"Consortiums\":{\"groups\": {\"$CONSORTIUM_NAME\":
```

应用通道更新

Then, follow the steps at Step 3: Re-encode and submit the config.

编辑节点组织

As stated above, these changes will have to be proposed by a system channel admin and sent to the relevant peer organization for signature.

我们需要针对所有应用通道上的组织，执行一系列类似的编辑。

Application channel updates

需要说明的是，不像系统通道，节点组织能够针对应用通道进行配置更新生成请求。如果你是针对你自己的组织进行配置变更，将不需要其他企业组织的签名确认。 但是，如果你要对其他组织进行配置变更，则那个组织将需要批准这次变更。

Edit the peer organizations

你需要导出以下变量:

We need to perform a similar set of edits to all of the organizations on all application channels.

- `CH_NAME`： 被更新的应用通道名称。
- `ORGNAME`： 你正在更新的组织名称。

- `TLS_ROOT_CA`： 排序节点TLS证书的绝对路径。
- `CORE_PEER_MSPCONFIGPATH`： 代表组织的MSP的绝对路径。
- `CORE_PEER_LOCALMSPID`： 提交的通道更新组织的MSP ID。这是节点服务组织中一员的MSP。
- `ORDERER_CONTAINER`： 一个排序节点容器的名称。当定位排序服务时，你可以在排序服务中定位任意一个特定节点。你的请求将会被自动推送至领导者。

环境变量设置完毕后，导航至 步骤 1： 拉取和翻译配置。

Note that unlike the system channel, peer organizations are able to make configuration update requests to application channels. If you are making a configuration change to your own organization, you will be able to make these changes without needing the signature of other organizations. However, if you are attempting to make a change to a different organization, that organization will have to approve the change.

`modified_config.json` 设置完毕后, 添加生命周期组织策略（比如 `enable_lifecycle.json` 文件中所展示的）使用以下命令：

You will need to export the following variables:

```
jq -s ".[0] * {\"channel_group\":{\"groups\":{\"Application\": {\"groups\": {\"$ORGNAME\": {\"pol
```

- `CH_NAME`: the name of the application channel being updated.
- `ORGNAME`: The name of the organization you are currently updating.
- `TLS_ROOT_CA`: the absolute path to the TLS cert of your ordering node.
- `CORE_PEER_MSPCONFIGPATH`: the absolute path to the MSP representing your organization.
- `CORE_PEER_LOCALMSPID`: the MSP ID of the organization proposing the channel update. This will be the MSP of one of the peer organizations.
- `ORDERER_CONTAINER`: the name of an ordering node container. When targeting the ordering service, you can target any particular node in the ordering service. Your requests will be forwarded to the leader automatically.

紧接着, 按照步骤[步骤 3： 重编码和提交配置](./config_update.html#step-3-re-encode-and-submit-the-config)。

Once you have set the environment variables, navigate to Step 1: Pull and translate the config.

编辑应用通道

Then, add the lifecycle organization policy (as listed in `enable_lifecycle.json`) to a file called `modified_config.json` using this command:

待所有应用通道更新至2.0更新包含 V2_0 的能力，新链码生命周期的背书策略必须被添加到每个通道。

```
jq -s ".[0] * {\"channel_group\":{\"groups\":{\"Application\": {\"groups\": {\"$ORGNAME\": {\"pol
```

你可以设置和你更新节点组织时设置的一样的环境变量。在这种情况下，需要注意的是你不需要更新配置中的组织配置，所以不需要使用 `ORGNAME` 变量。

Then, follow the steps at Step 3: Re-encode and submit the config.

环境变量设置完毕后，导航至 Step 1: 拉取和翻译配置。

Edit the application channels

`modified_config.json` 设置完毕后, 添加通道背书策略（比如 `enable_lifecycle.json` 文件中所展示的）使用以下命令：

After all of the application channels have been updated to include V2_0 capabilities, endorsement policies for the new chaincode lifecycle must be added to each channel.

```
jq -s '.[0] * {"channel_group":{"groups":{"Application": {"policies": .[1].appPolicies}}}}' confi
```

You can set the same environment you set when updating the peer organizations. Note that in this case you will not be updating the configuration of an org in the configuration, so the `ORGNAME` variable will not be used.

紧接着, 按照步骤Step 3: 重编码和提交配置进行操作。

Once you have set the environment variables, navigate to Step 1: Pull and translate the config.

若想要通道更新被批准，需要满足修改配置中 `Channel/Application` 部分的策略。默认情况下，这是通道中节点组织的 `MAJORITY`。

Then, add the lifecycle organization policy (as listed in `enable_lifecycle.json`) to a file called `modified_config.json` using this command:

编辑通道访问控制列表（可选）

```
jq -s '.[0] * {"channel_group":{"groups":{"Application": {"policies": .[1].appPolicies}}}}' confi
```

对于新生命周期，下面的 访问控制列表（ACL） 在 `enable_lifecycle.json` 展示的是默认值，但是你可以根据你的用例选择是否更改他们。

Then, follow the steps at Step 3: Re-encode and submit the config.

```
"acls": {
 "_lifecycle/CheckCommitReadiness": {
   "policy_ref": "/Channel/Application/Writers"
 },
 "_lifecycle/CommitChaincodeDefinition": {
   "policy_ref": "/Channel/Application/Writers"
 },
 "_lifecycle/QueryChaincodeDefinition": {
   "policy_ref": "/Channel/Application/Readers"
 },
 "_lifecycle/QueryChaincodeDefinitions": {
   "policy_ref": "/Channel/Application/Readers"
```

For this channel update to be approved, the policy for modifying the `Channel/Application` section of the configuration must be satisfied. By default, this is a `MAJORITY` of the peer organizations on the channel.

您可以保留与以前编辑应用程序通道时相同的环境。

Edit channel ACLs (optional)

环境变量设置完毕后，导航至 Step 1: Pull and translate the config.

The following Access Control List (ACL) in `enable_lifecycle.json` are the default values for the new lifecycle, though you have the option to change them depending on your use case.

`modified_config.json` 设置完毕后, 添加 ACL（比如 `enable_lifecycle.json` 文件中所展示的）使用以下命令：

```
"acls": {
  "_lifecycle/CheckCommitReadiness": {
    "policy_ref": "/Channel/Application/Writers"
  },
  "_lifecycle/CommitChaincodeDefinition": {
    "policy_ref": "/Channel/Application/Writers"
  },
  "_lifecycle/QueryChaincodeDefinition": {
    "policy_ref": "/Channel/Application/Readers"
  },
  "_lifecycle/QueryChaincodeDefinitions": {
    "policy_ref": "/Channel/Application/Readers"
```

```
jq -s '.[0] * {"channel_group":{"groups":{"Application": {"values": {"ACLs": {"value": {"acls": .
```

You can leave the same environment in place as when you previously edited application channels.

紧接着, 按照步骤[Step 3: 重编码和提交配置]进行操作(./config_update.html#step-3-re-encode-and-submit-the-config)。

Once you have the environment variables set, navigate to Step 1: Pull and translate the config.

若想要通道更新被批准，需要满足修改配置中 `Channel/Application` 部分的策略。默认情况下，这是通道中节点组织的 `MAJORITY` 。

Then, add the ACLs (as listed in `enable_lifecycle.json`) and create a file called `modified_config.json` using this command:

在 `core.yaml` 使用新生命周期

```
jq -s '.[0] * {"channel_group":{"groups":{"Application": {"values": {"ACLs": {"value": {"acls": .
```

若按照推荐流程 中使用一个像 `diff` 的工具以二进制来对比 `core.yaml` 的新版与旧版, 你将不需要将 `_lifecycle: enable` 加入白名单因为新的 `core.yaml` 已经加入到了 `chaincode/system` 。

Then, follow the steps at Step 3: Re-encode and submit the config.

但是，如果你直接更新旧节点的 YAML 文件，你需要将 `_lifecycle: enable` 添加至系统链码白名单。

For this channel update to be approved, the policy for modifying the `Channel/Application` section of the configuration must be satisfied. By default, this is a `MAJORITY` of the peer organizations on the channel.

如果想了解关于节点升级的更多信息，请参考升级你的组件

Enable new lifecycle in `core.yaml`

If you follow the recommended process for using a tool like `diff` to compare the new version of `core.yaml` packaged with the binaries with your old one, you will not need to add `_lifecycle: enable` to the list of enabled system chaincodes because the new `core.yaml` has added it under `chaincode/system`.

However, if you are updating your old node YAML file directly, you will have to add `_lifecycle: enable` to the list of enabled system chaincodes.

For more information about upgrading nodes, check out Upgrading your components.