configtxlator

`configtxlator` 命令允许用户在 protobuf 和 JSON 版本的 fabric 数据结构之间进行转换，并创建配置更新。该命令可以启动 REST 服务器并通过 HTTP 公开，也可以直接用作命令行工具。

语法

`configtxlator` 工具有五个子命令，如下：

- start
- proto_encode
- proto_decode
- compute_update
- version

configtxlator start

```
usage: configtxlator start [<flags>]

启动 configtxlator REST 服务

Flags:
  --help                 查看完整的帮助（可以尝试 --help-long 和 --help-man）。
  --hostname="0.0.0.0"   REST 服务器监听的主机名或者 IP。
  --port=7059            REST 服务器监听的端口。
  --CORS=CORS ...        允许跨域的域名，例如 '*' 或者 'www.example.com' （可能是重复的）。
```

configtxlator proto_encode

```
usage: configtxlator proto_encode --type=TYPE [<flags>]

将 JSON 文档转换为 protobuf。

Flags:
  --help                 查看完整的帮助（可以尝试 --help-long 和 --help-man）。
  --type=TYPE            要将 protobuf 编码成的类型。例如'common.Config'。
  --input=/dev/stdin     包含 JSON 文档的文件。
  --output=/dev/stdout   要将输出写入的文件。
```

configtxlator proto_decode

```
usage: configtxlator proto_decode --type=TYPE [<flags>]

将 proto 消息转换为 JSON。

Converts a proto message to JSON.

Flags:
  --help                 查看完整的帮助（可以尝试 --help-long 和 --help-man）。
  --type=TYPE            要将 protobuf 解码成的类型。例如'common.Config'。
```

```
  --input=/dev/stdin    包含 proto 消息的文件。
  --output=/dev/stdout  要将 JSON 文档写入的文件。
```

configtxlator compute_update

```
usage: configtxlator compute_update --channel_id=CHANNEL_ID [<flags>]

比较两个编码（marshaled）的 common.Config 信息，并计算它们的更新。


Flags:
  --help                    查看完整的帮助（可以尝试 --help-long 和 --help-man）。
  --original=ORIGINAL       原始配置信息。
  --updated=UPDATED         更新的配置信息。
  --channel_id=CHANNEL_ID   本次更新的通道名。
  --output=/dev/stdout      要将 JSON 文档写入的文件。
```

configtxlator version

```
usage: configtxlator version

显示版本信息。

Flags:
  --help  查看完整的帮助（可以尝试 --help-long 和 --help-man）。
```

示例

解码

将一个名为 `fabric_block.pb` 的块解码为 JSON 并打印到标准输出。

```
configtxlator proto_decode --input fabric_block.pb --type common.Block
```

或者，在启动 REST 服务器之后，使用下面的 curl 命令通过 REST API 执行相同的操作。

```
curl -X POST --data-binary @fabric_block.pb "${CONFIGTXLATOR_URL}/protolator/decode/common.Block"
```

编码

将策略的 JSON 文档从标准输入转换为名为 `policy.pb` 的文件。

```
configtxlator proto_encode --type common.Policy --output policy.pb
```

或者，在启动 REST 服务器之后，使用下面的 curl 命令通过 REST API 执行相同的操作。

```
curl -X POST --data-binary /dev/stdin "${CONFIGTXLATOR_URL}/protolator/encode/common.Policy" > po
```

Pipelines

从 `original_config.pb` 和 `modified_config.pb` 计算配置更新，然后将其解码为 JSON，再将其打印到标准输出。

```
configtxlator compute_update --channel_id testchan --original original_config.pb --updated modifi
```

或者，在启动 REST 服务器之后，使用下面的 curl 命令通过 REST API 执行相同的操作。

```
curl -X POST -F channel=testchan -F "original=@original_config.pb" -F "updated=@modified_config.p
```

附加笔记

该工具名称是 *configtx* 和 *translator* 的组合，旨在表示该工具只是在不同的等效数据表示之间进行转换。它不生成配置。它不提交或检索配置。它不修改配置本身，只是在 configtx 格式的不同视图之间提供一些双射操作。

对于 REST 服务器，既没有配置文件 `configtxlator`，也没有包含任何身份验证或授权工具。因为 `configtxlator` 不能访问任何可能被认为敏感的数据、关键材料或其他信息，所以服务器的所有者将其暴露给其他客户端没有风险。但是，由于用户发送到 REST 服务器的数据可能是机密的，所以用户应该信任服务器的管理员、运行本地实例或者通过 CLI 进行操作。