**Section 6: Configtxgen tool**

**Introduction to the configtxgen tool**

1. configuration artifacts that can be generated by way of the config gen tool.
2. Overview of the tool

* The files I have used in the demos for this section are available under the configtx

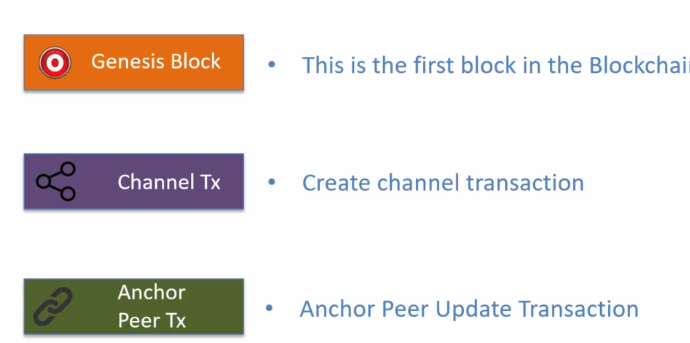


* all commands will be executed under this folder in the terminal

configtx gen is a command line utility for managing configuration artifacts.

There are **three types of artifacts** that you can manage with config gen.

1. **genesis block->** which is the order channel block used by the order binary.
2. **Create channel transaction->**So when you have to create an application channel, you can use the config gen to create the channel transaction.
3. **anchor peer update transaction->**This is for managing the anchor peers for the organizations that are part of the network.

****

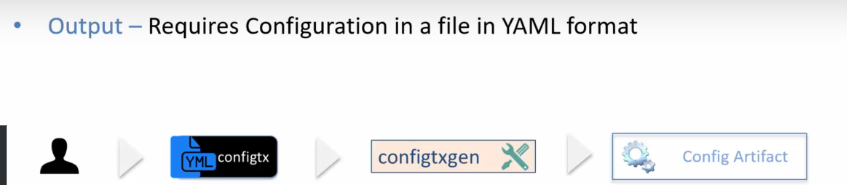
Please note that the third artifact type is not needed in fabric 2.0.

As a result, the configtxgen option for generating the peer update transaction is now deprecated It is still supported by the tool, but in later versions it may be removed.

The tool exposes **two type of commands:**

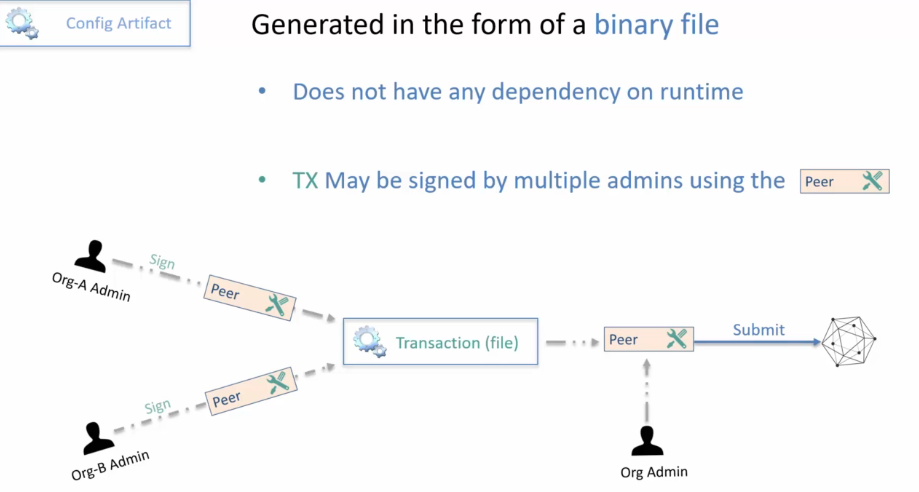
1. **output commands->**These output commands generate the configuration artifact.

* These commands require you to provide the configuration parameters in a file which is in Yaml format.
* So as a developer or as an administrator, you would put together a config.yaml file which will have all the parameters and the values.
* This file is used as an input to the config gen tool and that generates the required configuration artifact.



1. **inspection commands->**outputs configuration as JSON

* The generated configuration artifacts are files.
* They are in **binary format**, so you cannot open those files in a text editor
* in order to review the information contained in the configuration artifact files, you need to convert them to Json.
* And to do that you would use the inspection commands exposed by configtxgen utility.
* The configtxgen tool does not have any dependency on the runtime.
* In other words, to execute the configtxgen command, you don't need to have the fabric network up and running,
* No orders or peers are connected to by the config gen tool.
* the config gen tool simply generates the files and these files are in binary format.
* Now the one question that may occur to you at this point is why is it that the config gen is not interacting with the fabric runtime to execute these transactions?
* And the reason is that sometimes policies associated with the network govern the number of signatures that are needed for executing a transaction.
* For example, if the policy states that addition of organization requires organization A and organization B both to sign, then in that case the transaction generated by the config gen tool will need to be signed by the two organizations.
* So to illustrate this scenario, you will create the transaction file and then you will have the org A admin and org B admin sign that transaction file and then one of the admins will submit that transaction to the network for execution.



* The configtxgen tool is a command line tool.
* That is, there is no graphical interface to the tool.
* You would execute the configtxgen commands on the command shell window or the terminal window.
* The format of the command is configtxgen, you provide the command name and then you provide some optional flags and arguments
* to get help on configtxgen tool you can use the command dash help and to get the version you can use the command dash version.

1. command configtx version and you get the version information and
2. to get the help simply use dash help.

**Command to run in terminal->**

1. configtx version

2. configtx -help

In this section you will learn about:

how to set up the configtx.yaml file and then

all the commands on the configtxgen tool for generating the various configuration artifacts such as **the Genesis block, create channel transaction, anchor peer update transaction**

you would be in a position to create your own configurations using the config gen tool.

**Setting up the configtx.yaml file**

Learn:

1. config file setup
2. how the config gen tool uses the config yaml file.

The config dot yaml file contains the values for the various configuration elements for the artifacts that can be generated from the configtxgen tool.

The config file path is specified by way of an **environment variable**

**FABRIC\_CFG\_PATH**

* So this environment variable points to the folder that contains the confitx.yaml file.
* If this variable is not set, then the configtxgen tool searches for config.yaml file in the current folder and that's where it picks up the configuration information from.

**Properties may be overridden by setting Environment variable**

The way it works is that you would set up the shell with the parameters for which you want to override the values.

For example, if you want to override the **order type**, let's say the order type is set to solo in the configtx Yaml file and you want to set it as Kafka.

You can do so by using the

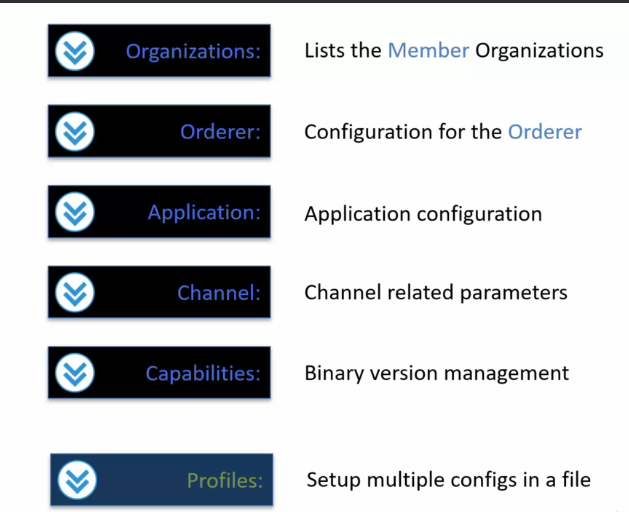
**Export CONFIGTX\_ORDERER\_ORDERERTYPE=kafka**

So this way you can override any property specified in the configtx.yaml file.

My suggestion would be not to do this if possible because then your configuration artifacts will be more dependable and predictable because you're not overriding anything in the configtx yaml file.

But again, when you are testing this may be a good way to make changes to test out the various configurations.

Config yaml file has six sections.



This profile section will become clear once you will see how configtxgen tool is used with the configtx yaml file.

**Sample Setup**

In this sample setup there will be:

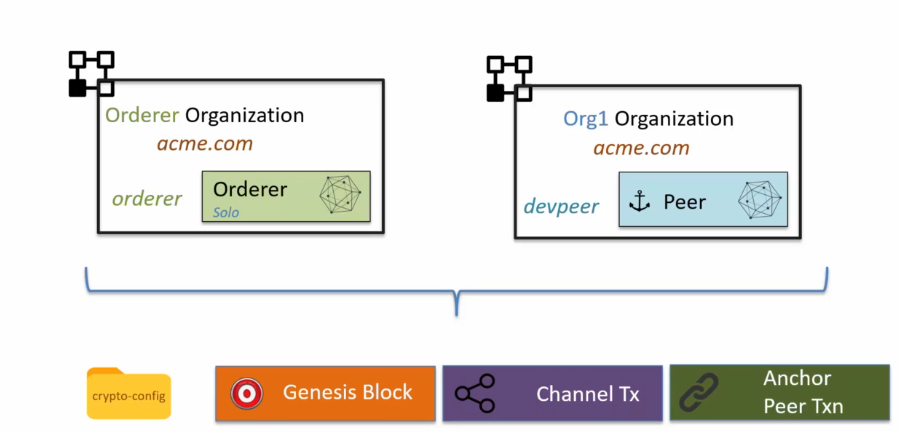
1. one order of organization.

* The name of the organization is order and then there will be

1. one peer organization.

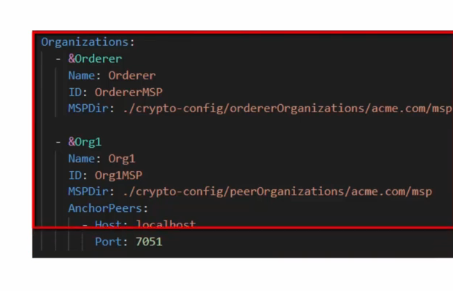
* The name of that organization is org1.

1. There will be an order of type solo.
2. There will be one **anchor peer** within the org1, one organization referred to by the name **Dev Peer**.
3. Then there will be a crypto config folder that we will have to create by using the crypto gen utility as the organization configuration requires access to the MSP that is the crypto material.



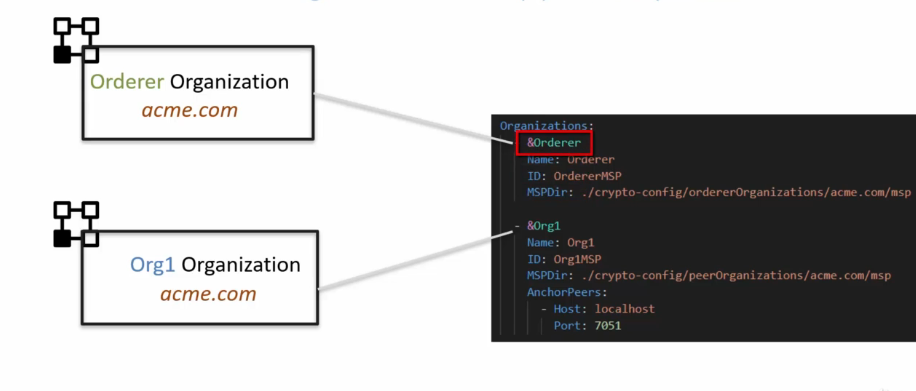
**organization section:**

1. This is the key section where a list of member organization is managed.
2. Now one thing that you will realize once you will start to go through the configuration is that references to the organizations is made from the other sections, from the application section, from the order section, as well as from the profile section.
3. So to avoid repeating the organization details, this section lists out the organization in the form of Yaml anchors.
4. Here is how the organization section looks like in the config yaml file.

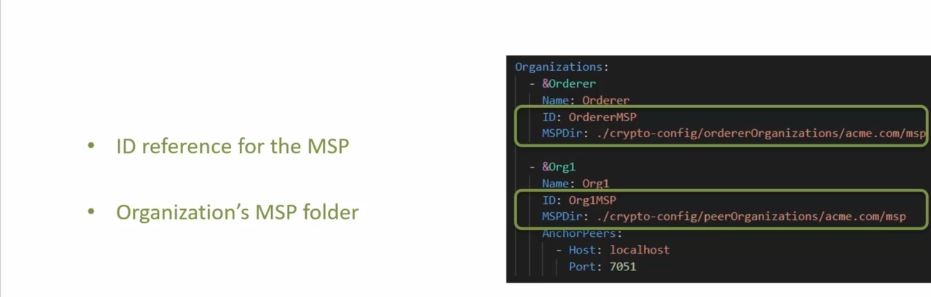




* We have two organizations, the **order organization** and **the org one organization.**
* Each of these organizations have been defined as anchor elements.
* What that means is that they can be referred to from other parts of the config yaml file.



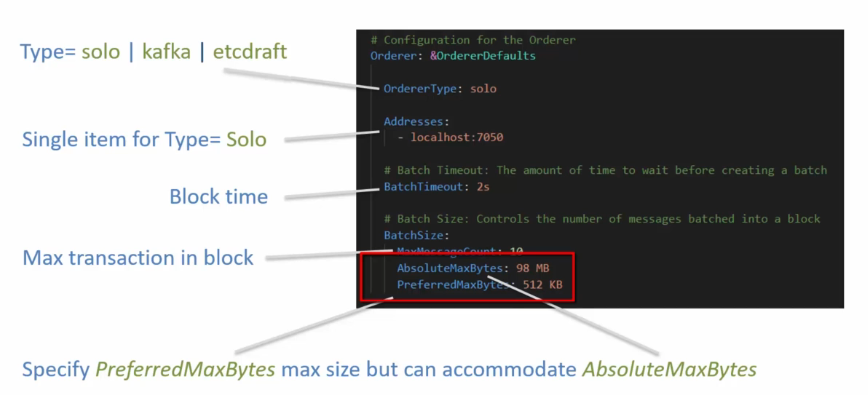
* Each organization has a name, then there is an ID, an MSP directory.
* ID is an ID reference for the MSP for the organization.
* It identifies the organization.
* Then there is an MSP directory which is a parameter that points to the MSP folder for the organization.
* This MSP folder has the crypto material for the organization



* the peer organization such as org 1 Also define a list of anchor peers in the organization Definition.
* These anchor peers are known in the network.
* If there is a new peer that's added to the organization's Infrastructure and is not part of the network configuration, then that peer is not known outside the organization.

**orderer section:**

* order section is also defined as an anchor element so that you can refer to it from other sections within the configtx yaml file.
* The order type element may be **solo Kafka or Etcdraft**.
* In the test setup it is **set to solo**.
* The address element is a list of addresses for the order.
* In the case of Solo, there can only be one address which is set to **localhost:7050**.
* If it was Kafka or Raft, there will be multiple addresses.
* The next one is the **batch timeout**, which basically decides the block time for the network.
* That is how long to wait after the first transaction before a block is getting created.
* The next set of attributes are managed under the heading **batch size**.
* The max message count, as the name indicates, is the maximum number of messages that can be in a block.
* Then there are these two attributes the preferred max bytes and the absolute max bytes.



Let me explain how this works.

1. The administrator or the developer of the configuration can specify the preferred max bytes maximum size, but the block can accommodate the size specified by absolute max bytes.

**Example:**

* let's say the order has received six transactions.
* Now these six transactions are of varying sizes.
* Let's say the first three transactions collectively are 512 KB.
* In that case, once the order reaches that 512 KB of preferred max byte size, the order will seal the
* block and send it out.
* The remaining transactions will be sent out in a next block.
* Now, it is very much possible that in the next block, the one transaction itself was like 50 MB.
* In that case, the preferred max byte size has exceeded, but it is still under the absolute max byte size.
* So the block will be created with that one transaction and sent out to the peers.

**Application section**

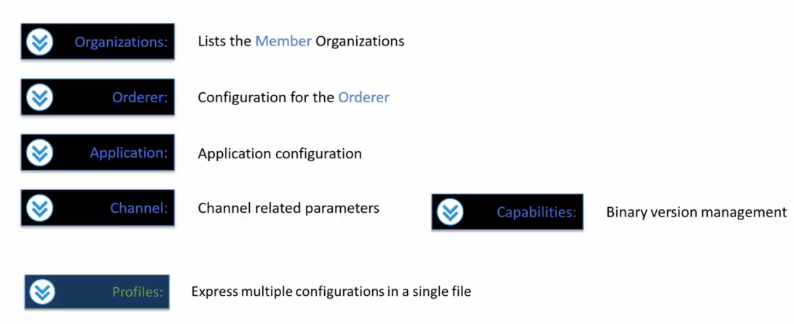
* The application section has a **subsection organizations** under which there is a list of organizations that participate in the application type transactions.

**Channel section**

* The channel section defines the default set of parameters for the channel.
* It sets the default policies.

**SUMMARY**

1. talked about configtx yaml file which is used by the configtx gen tool
2. in order for configtxgen tool to access the config yaml file you need to set up the **FABRIC\_CFG\_PATH** environment variable to point to the location of the config yaml file.
3. There are six sections in the config yaml file.



**Configtx YAML Profiles Section**

Discuss:

1. the profiles section and
2. how profiles are used by the configtxgen tool.

There can be multiple profiles declared in a config yaml file.

Each of these profiles are named subsections under the profiles section and

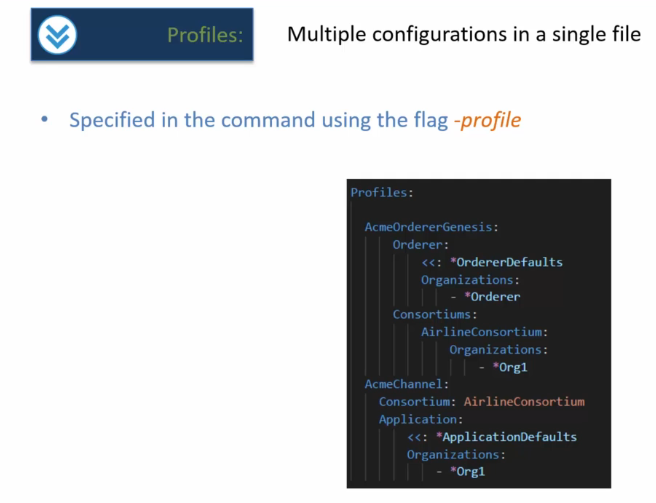
each of these profiles are needed for generation of specific configuration components.

The attribute under each of the profiles depend on the configuration component type.

So **for example**, the Genesis block profile will look different compared to a application channel profile.

values for the attributes or parameters for each of the profiles Refer to the elements in other sections of config Yaml file.

So this is the sample profiles section:

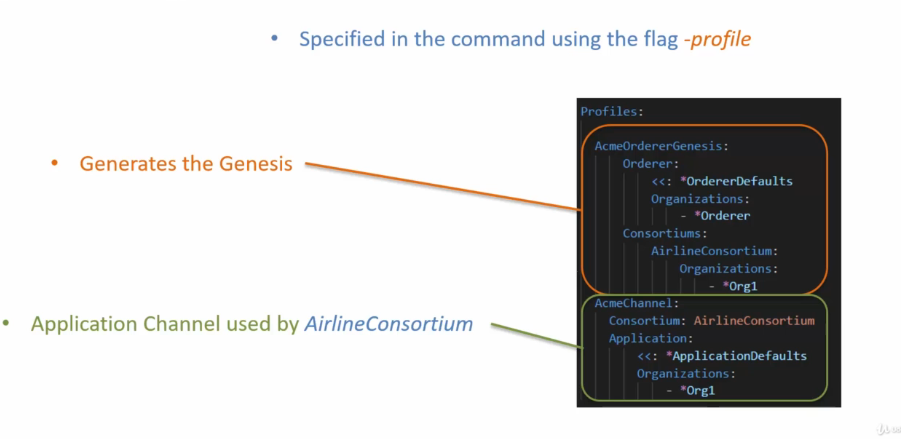


Herein you will see that there are two profiles

* Acme Orderer, Genesis and
* Acme Channel.

The **Acme Orderer Genesis** profile is for creating the Genesis block and

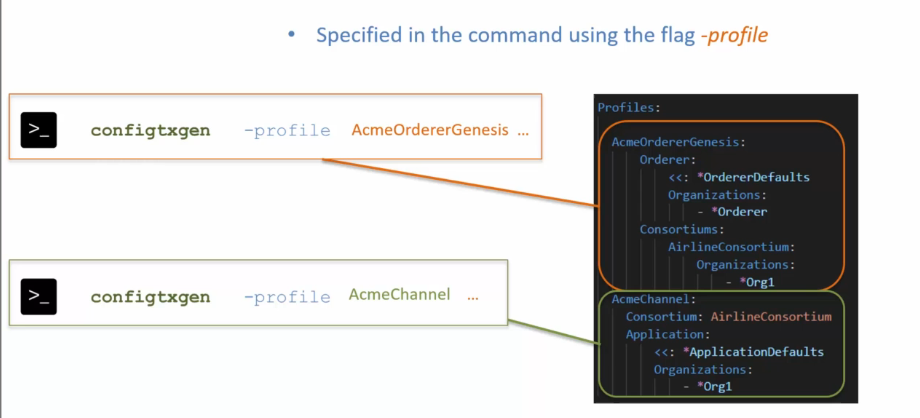
the **Acme Channel profile** is for creating the Create Channel transaction.



Each of these, you will notice, have a different set of attributes or sections as they generate different type of configuration artifacts.

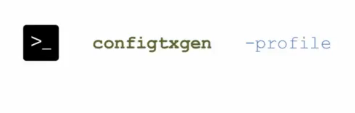
These named profiles are used from the config gen tool by way of the dash profile flag.

So here is an example of how the config gen tool will be accessing the profile Acme or Genesis.And then similarly, in order to access the Acme Channel profile, we would set **the -profile flag value to Acme Channel**.



**SUMMARY**

* talked about the profile section in configtxyaml file.
* You can create multiple profiles in the configtxyaml file and these profiles represent multiple configurations for different type of network configurations.
* References are made to other parts of the configtx yaml file from the various profiles under the profiles section.
* Profile is specified to the config gen tool by using the flag dash profile.

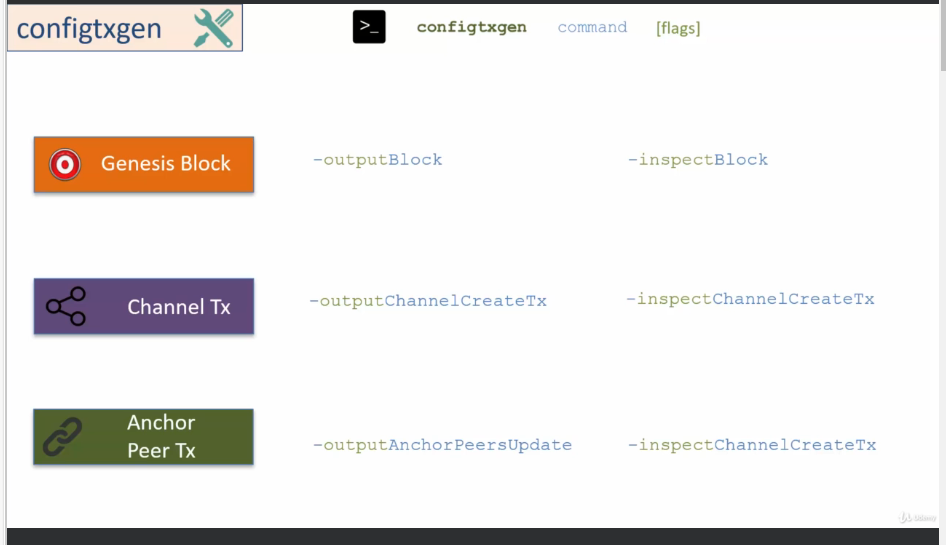


**Generating the Genesis Block**

Objectives:

1. you should be able to describe the configuration elements that you need to set up in the configtx Yaml file for generating the genesis Block
2. you will know the commands that you can use for creating the Genesis block file and for inspecting the Genesis block file.

Let's go over the configtxgen commands that are used for generating the network configuration artifacts for generating the genesis block.



**in this lecture is on the commands for creating the genesis block.**

**GENISIS BLOCK:**

The Genesis block generation requires certain configuration element from the configtx.yaml file.

* It requires the configuration for the order.
* It requires the configuration for the order organizations, that is the organizations that would host the order.
* Thus there is a need for the MSP for the organizations to be available.
* It also requires the configuration for the consortiums.
* Now consortiums have member organizations.
* Thus there is a need for the MSP, for the consortium organizations to be also available for the configtxgen tool to generate the Genesis block

for the walkthrough, I will use the crypto configtxYaml file that I created in the previous section.

In this configtxyaml file, there is

1. one organization for the order
2. peer organization with the name Org one.

The MSP or the crypto material will be generated by using crypto gen and the way it will work is we will pass this crypto config.yaml file to crypto gen which will generate the crypto material under the folder crypto -config.





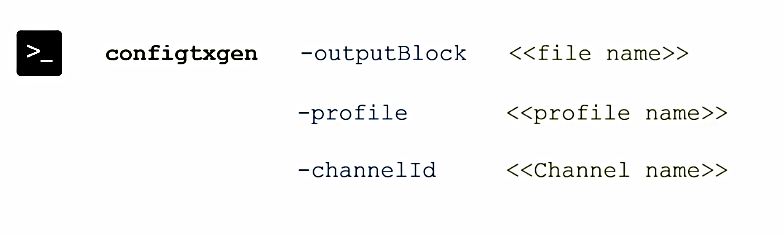


**Generate the genesis block:**

**Command**:

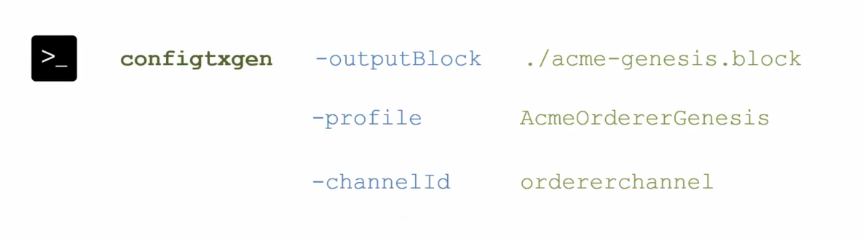


A profile has to be specified by way of the flag dash profile, and a channel ID has to be specified.



Now, if you do not specify the channel ID, in that case, it would generate the genesis block with a default channel ID test chain ID.

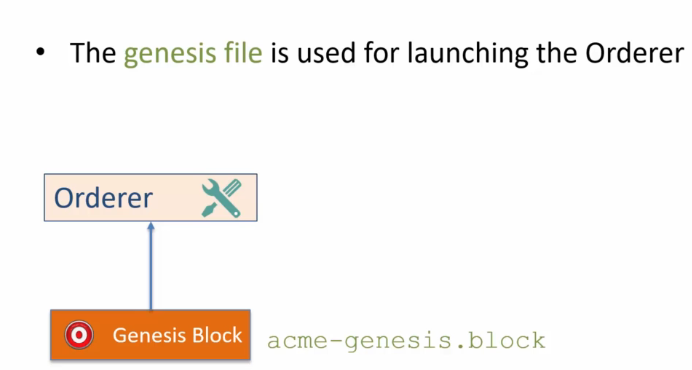
for the walkthrough the **command** will be:



After the successful execution, you will see a file Acme-Genesis.block generated in the current folder.

An obvious question that you may ask at this point is how is this generated file used now?

This generated file is used for launching the order.



The profile specified in the command Acme Order.

Genesis has all the configuration elements that are needed for the generation of the Genesis block.

So here we have the order which is pulled from the order defaults.

Now if you look at.

The config yaml file, you will find that the order default maps to these configuration elements.

The order type is solo.

The address of the order is localhost colon 7050.

And then there are values for the batch timeout and the batch size.

Next, the Genesis block requires the order organizations which are coming from here.

The anchor order in the config yaml file maps to this and you will find that there is an order MSP which

is the ID of the orders MSP and there is a reference to the MSP directory that points to the crypto

config folder.

The last one is the consortiums.

In the consortiums we have a list of member organizations that are part of the consortium.

In this case it's just org one, which is the peer organization.

Let's go ahead and run the command config txn dash output block Acme Dash.

Genesis dot block Dash profile Acme Order Genesis Dash Channel ID Order Channel.

Once you execute this command, you will see that a new file is generated.

Acme Dash Genesis Dot Block.

Keep in mind this is a binary file so you cannot open in an editor, but you can inspect it.

Let me talk about the command that you can use for inspecting the Genesis block.

You can dump the content of the Genesis block in Json format by using the command config gen dash inspect

block and you have to provide the file name, which in this case is Acme Dash, Genesis Dot block.

And once you run this command, it will print out the content of the block in Json format.

So config gen dash inspect, block, acme dash, genesis, dot block.

And once you do that, as you can see it dumped out the Json on the terminal, which is not as useful.

So what I'm going to do is I'm going to create a folder under simple to org, call it the temp folder,

run the same command and just pass the generated Json to a file by the name Acme Dash.

Genesis dot Json under the temp folder.

So this will generate the Json file under the temp folder and you can inspect this file by using the

editor in Visual Studio code.

In this lecture you'll learn how to generate the Genesis block and how to inspect the generated Genesis

block file using the config gen tool to generate the Genesis block, you need to set up the config yaml

file that has the configuration for the genesis block and then you would use the config gen command

config gen dash output block to generate the genesis block, the generated file is in binary format,

so you cannot open it in an editor in order to inspect the generated Genesis block file, you need to

use the command inspect block.

This command prints out the block in Json format on the terminal, but you can direct the output of

this command to a file using the pipe test dot Json, for example, here, and then you can inspect

the content as Json in any Json editor.

Autoscroll

Course content

Overview

Q&AQuestions and answers

Notes

Announcements

Reviews

Learning tools

**Generating the Create Channel Transaction**

**Generating the Anchor Peer Update Transaction**

**Exercise-Setup a new Channel Profile with 2 Orgs**