**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

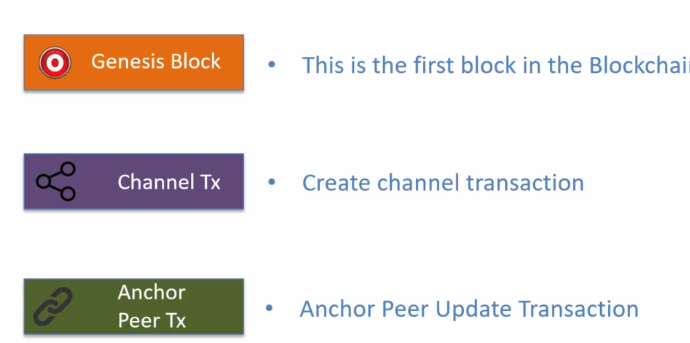


* all commands will be executed under this folder in the terminal

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

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

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.

an **artifact** is a file or data structure generated as part of setting up and managing the network. Artifacts contain configurations and information that the blockchain network components (like orderers and peers) need to operate correctly. They essentially "tell" the network how to organize, who the participants are, and the rules for interacting on specific channels.

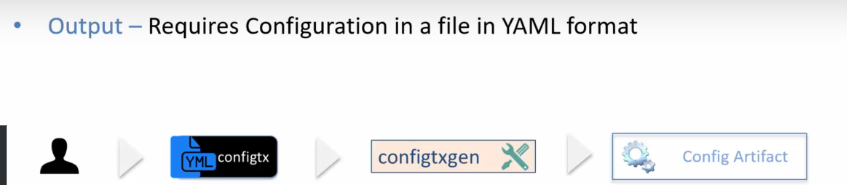
The tool exposes **two type of commands:**

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.

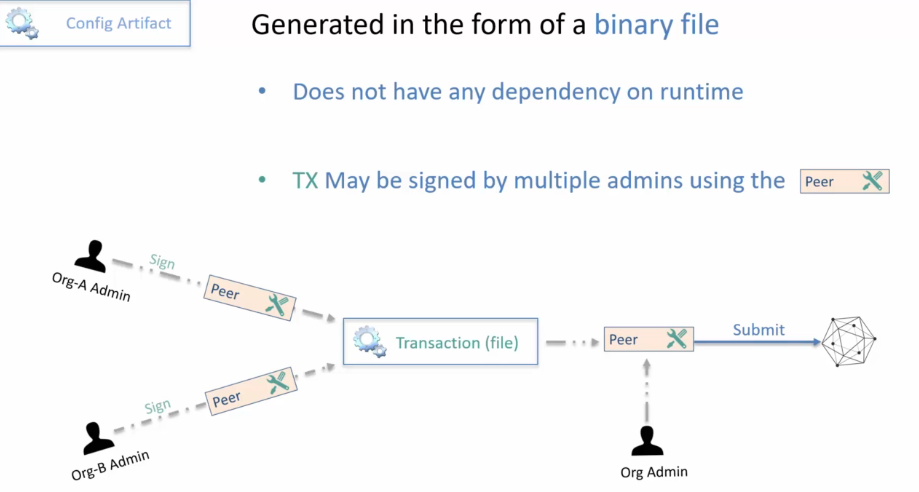
**Process**:

* You then use a tool called the **config generator tool**. This tool reads your config.yaml file and creates the configuration artifact based on the details you provided.
* Once created, the configuration artifact is ready for the system to use, making sure it operates with all the settings you specified.



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 configtxgen tool.
* the configtxgen 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 configtxgen 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 configtxgen 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.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

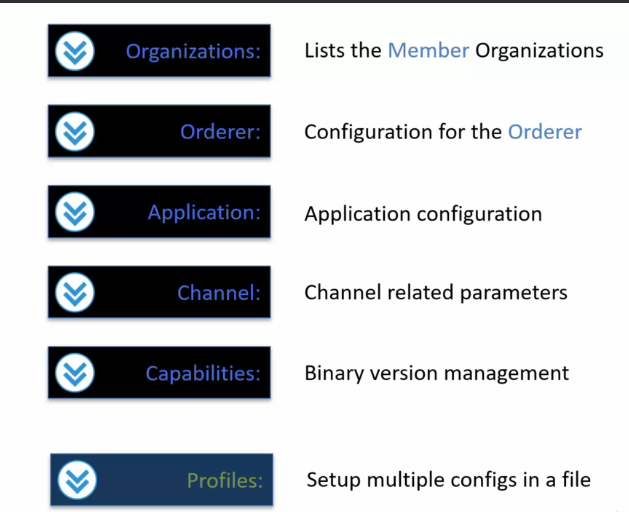
**Export CONFIGTX\_ORDERER\_ORDERERTYPE=kafka**

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:

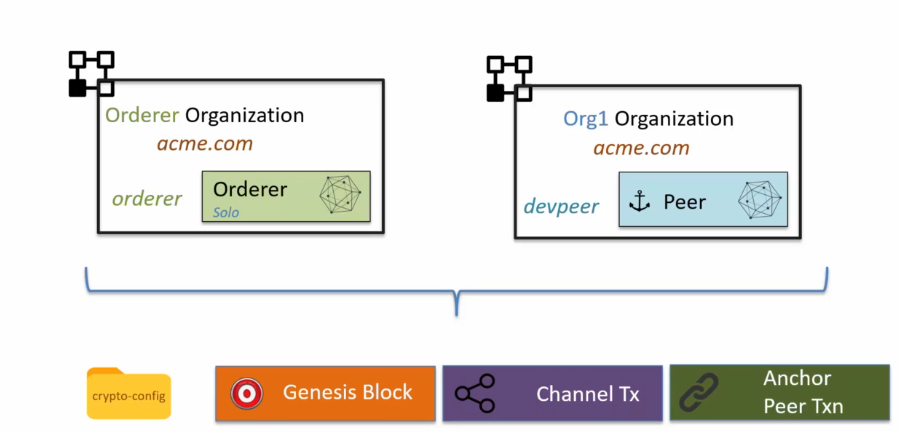
1. one order of organization.

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

1. one peer organization.

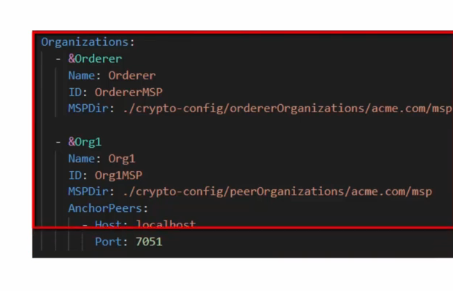
* The name of that organization is org1.

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.



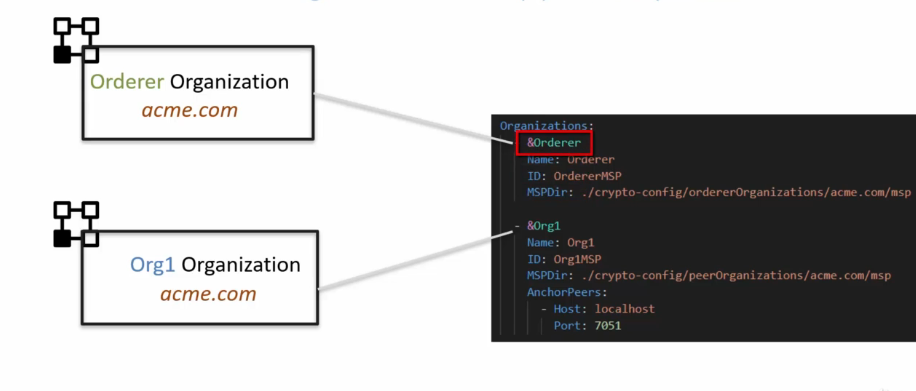
**organization section:**

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.





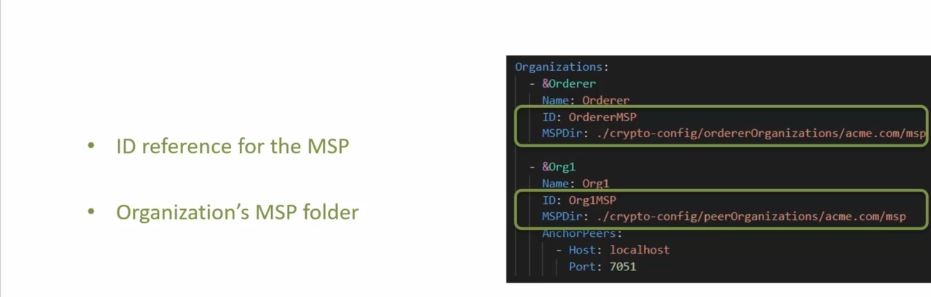
* We have two organizations, the **order organization** and **the org one organization.**
* Each of these organizations have been defined as anchor elements.
* What that means is that they can be referred to from other parts of the config yaml file.



* Each organization has a name, then there is an ID, an MSP directory.

The **MSP directory** (Membership Service Provider directory) is a crucial component for managing identities and permissions of members (organizations) within the blockchain network. It provides the cryptographic materials needed for identity validation and authentication.

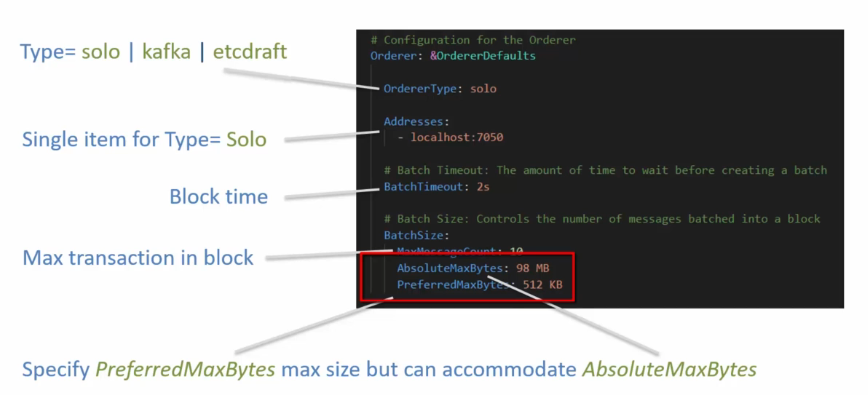
* ID is an ID reference for the MSP for the organization.
* It identifies the organization.
* Then there is an MSP directory which is a parameter that points to the MSP folder for the organization.
* This MSP folder has the crypto material for the organization



* 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**

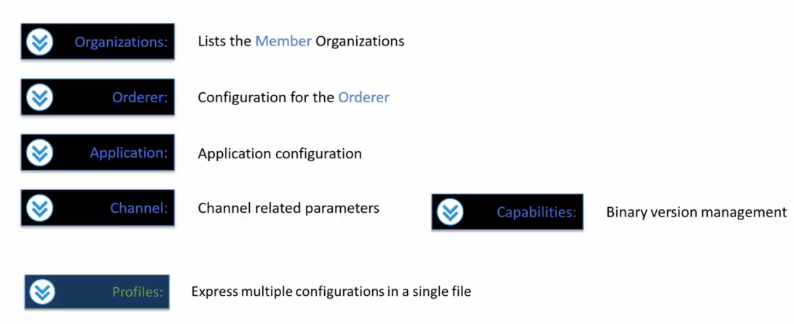
* The application section has a **subsection organizations** under which there is a list of organizations that participate in the application type transactions.

**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 configtxgen 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

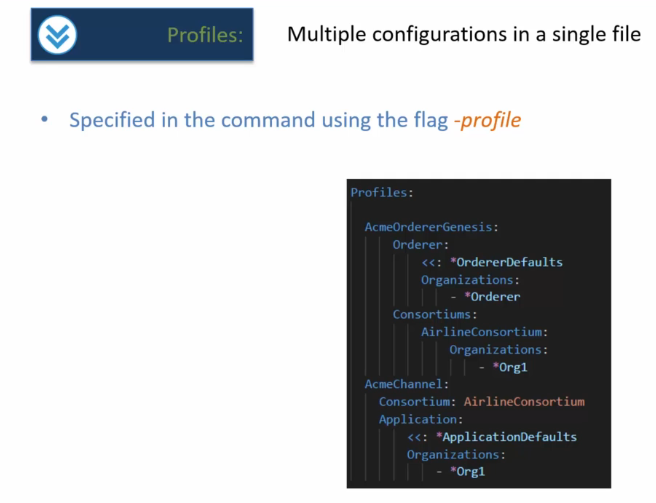
each of these profiles are needed for generation of specific configuration components.

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:

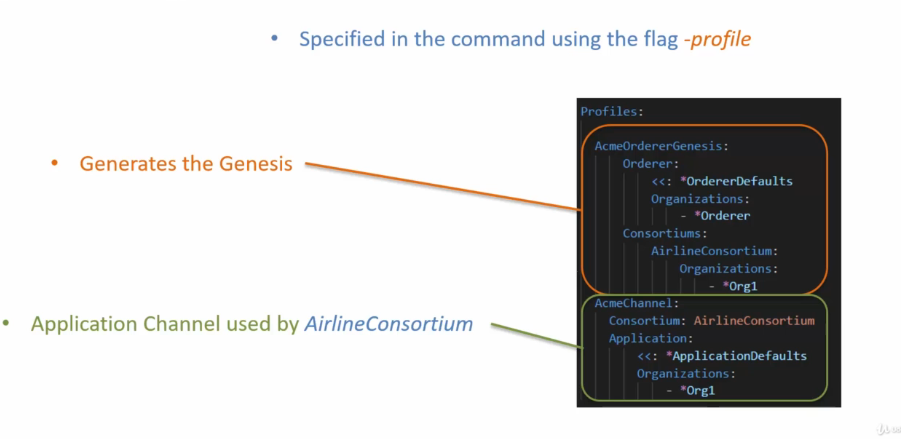


Herein you will see that there are two profiles

* Acme Orderer, Genesis and
* Acme Channel.

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

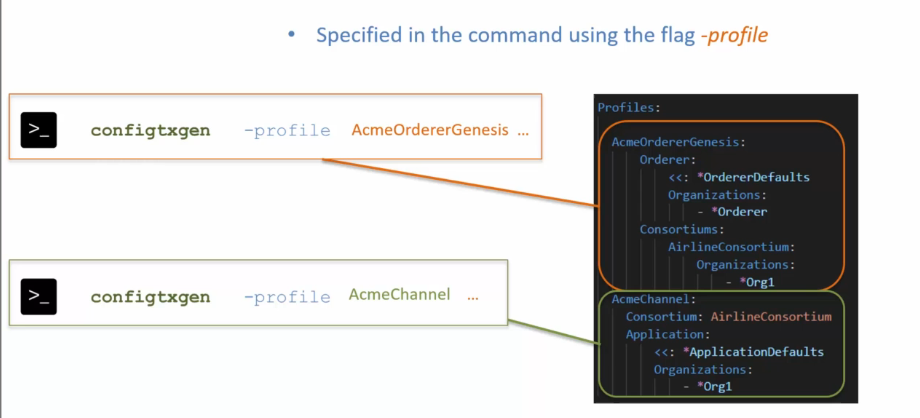
the **Acme Channel profile** is for creating the Create Channel transaction.



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

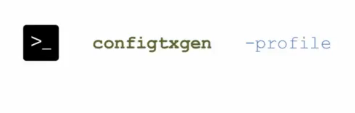
These named profiles are used from the config gen tool by way of the dash profile flag.

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.

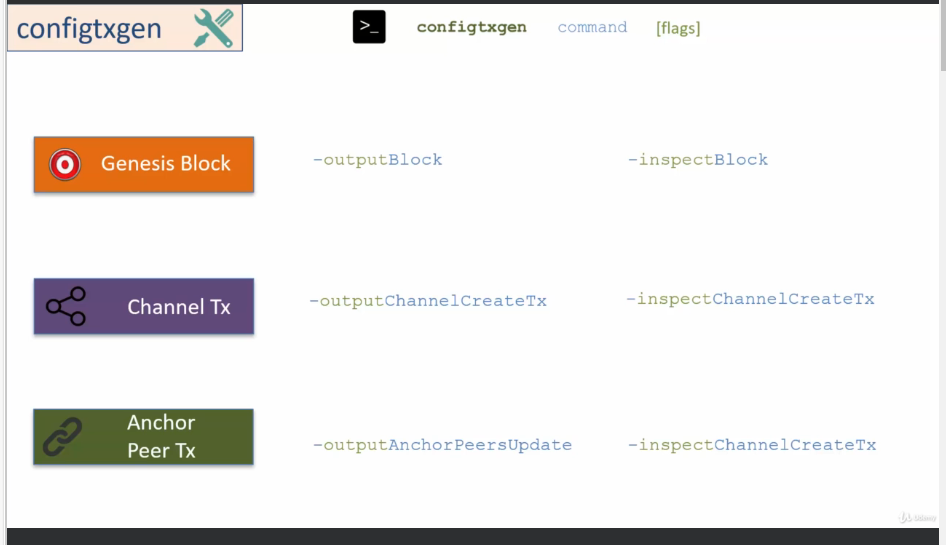


**Generating the Genesis Block**

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.





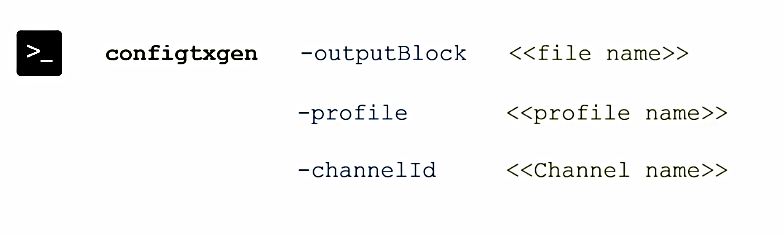


**Generate the genesis block:**

**Command**:

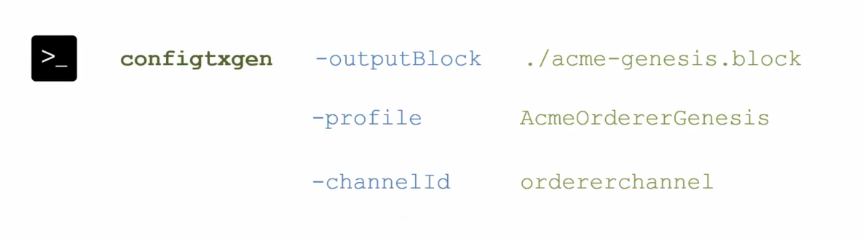


A profile has to be specified by way of the flag dash profile, and a channel ID has to be specified.



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.

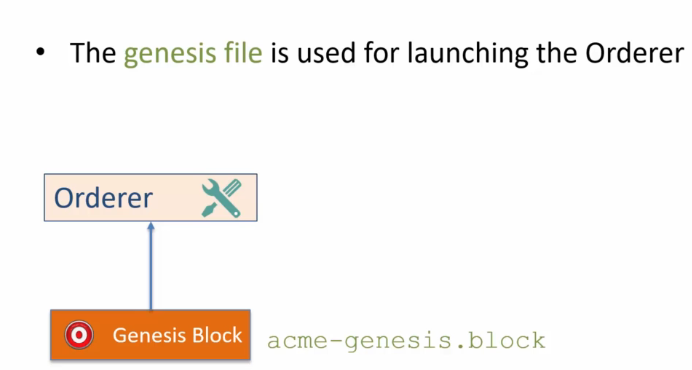
for the walkthrough the **command** will be:



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

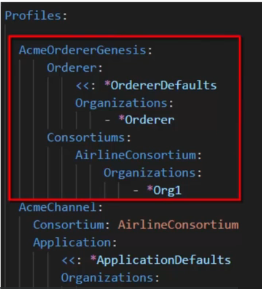
An obvious question that you may ask at this point is how is this generated file used now?

This generated file is used for launching the order.

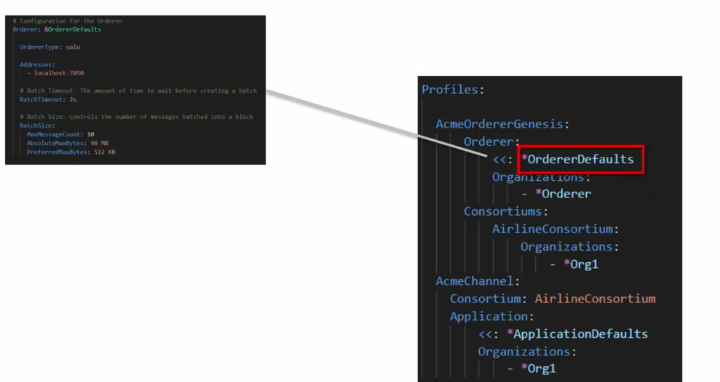


The profile specified in the command AcmeOrdergenesis.

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



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



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

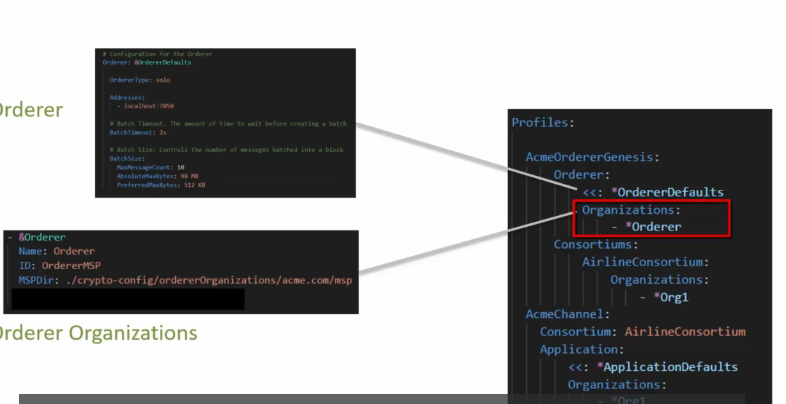
The order type is solo.

The address of the order is localhost colon 7050.

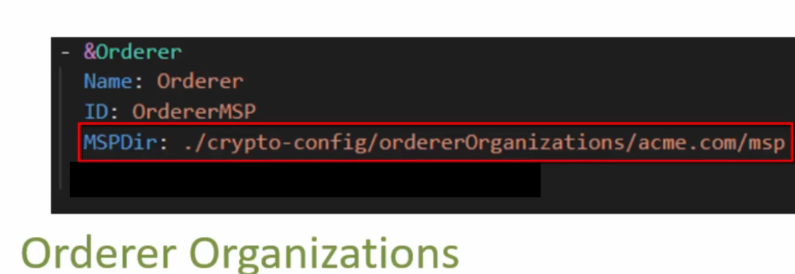
And then there are values for the batch timeout and the batch size.



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



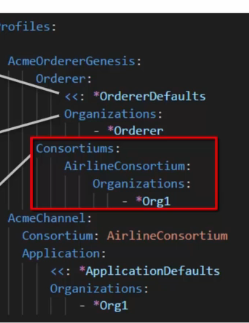
The anchor order in the config yaml file maps to this and you will find that there is an order MSP which is the ID of the orders MSP and there is a reference to the MSP directory that points to the crypto-config folder.



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.



**Command to run on terminal:**

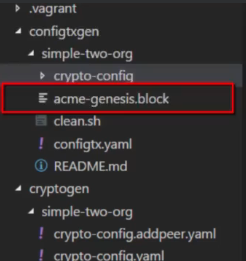
configtxgen -outputBlock ./acme-genesis.block -profile AcmeOrdererGenesis -channelID ordererchannel

**output:**



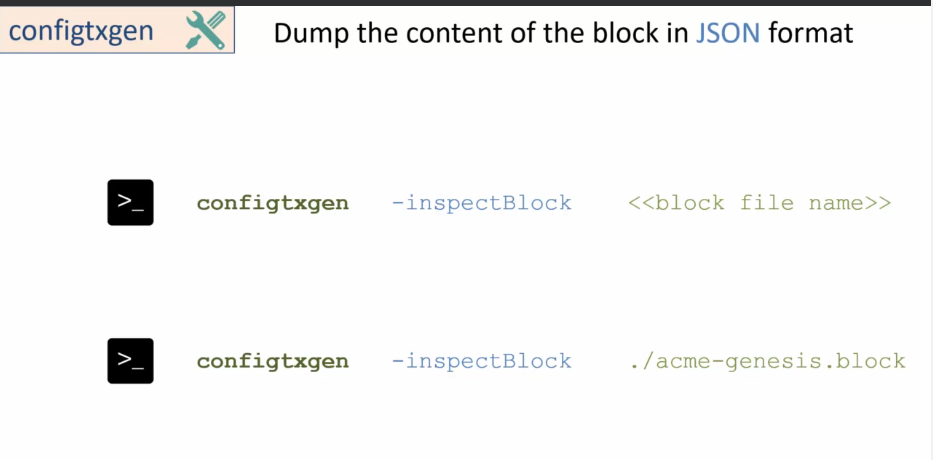
Once you execute this command, you will see that a new file is generated.

Acme-Genesis.block



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

**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 configtxgen -inspectBlock
* you have to provide the file name, which in this case is Acme-Genesis.block
* And once you run this command, it will print out the content of the block in Json format.

**Command to run on terminal:**

configtxgen -inspectBlock ./acme-genesis.block

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

Instead of getting on terminal ->

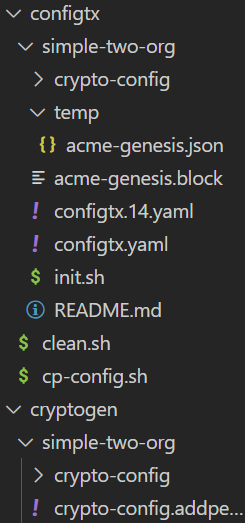
**Create folder under simple-two-org->temp**

run the same command and just pass the generated Json to a file by the name Acme-Genesis.Json under the temp folder.

**Command to run on terminal:**

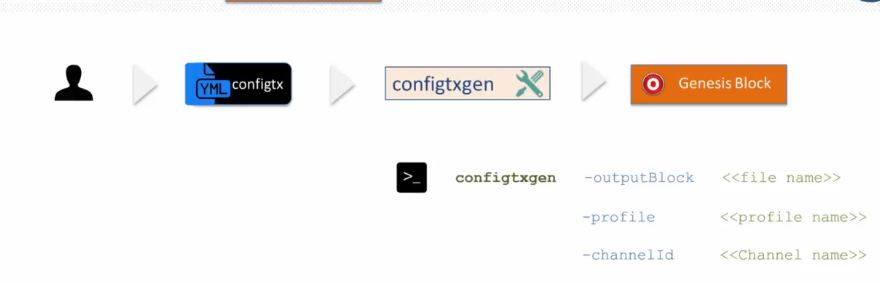
configtxgen -inspectBlock ./acme-genesis.block > temp/acme-genesis.json

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.



**SUMMARY**

1. how to generate the Genesis block and how to inspect the generated Genesis block file using the configtxgen tool
2. to generate the Genesis block, you need to set up the configtxyaml file that has the configuration for the genesis block and then you would use the configtxgen command **configtxgen -outputBlock** to generate the genesis block



1. 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



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 temp.Json/test.json, for example, here, and then you can inspect the content as Json in any Json editor.



**Generating the Create Channel Transaction**

you will learn about the

1. configuration elements that you need to set up in the config Yaml file for the Create channel transaction
2. the commands that you can use for creating the Create channel transaction file.

I'll cover two commands:

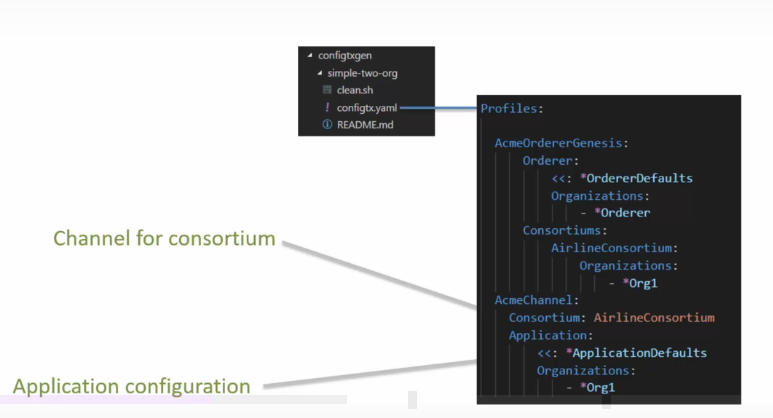


The channeltx creation requires you to provide certain configuration elements by way of the config Yaml file.

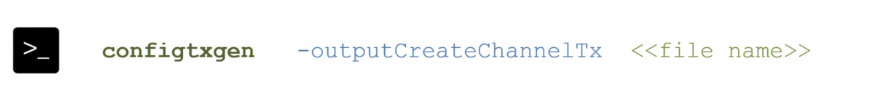
* It requires you to provide the configuration for the application.
* It requires the configuration for the **consortiums** and the consortiums have organization members.
* These members have associated **MSPs** that need to be accessible to the config gen tool.
* Now the members in this consortium will have access to the channel created by way of this channel transaction.
* It also requires the channel ID, the configuration element for the channel transaction are available under the profile Acme Channel.

So if you look at configtx. Yaml file, the channel has the channel consortium and then it has the application configuration.

And these are the two elements needed for the channel transaction



,to generate the application channel transaction you would use the **command**



As part of this you have to specify the file name that gets generated as a result of the execution of this **command.**



You have to specify the profile name and the channel ID.

The **command** that I'll use in the walkthrough will look like



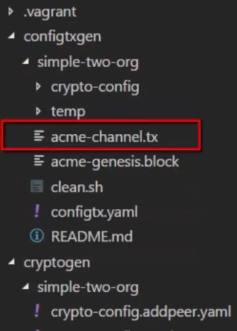
Execution of this command will result in the generation of file Acme-channel.tx in the current folder.

Let's go ahead and execute the command

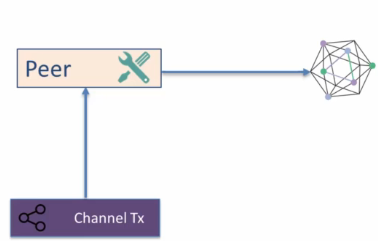
**Command to run on the terminal:**

**Configtxgen -outputCreateChannelTx ./acme-channel.tx -profile AcmeChannel -channelID acmechannel**

Once this command has executed, it will create the file:



This file is used by the peer binary for submitting a transaction for the creation of application channel.



The Create Channel transaction file is in binary format.

To inspect the content you will need to use the **command**:



You have to provide the name of the file for which you want to dump the content in Json format.



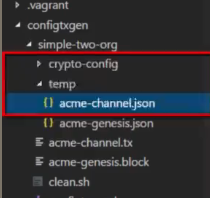
Let's execute the command

**Command to run on the terminal:**

configtxgen -inspectChannelCreateTx acme-channel.tx > ./temp/acme-channel.json

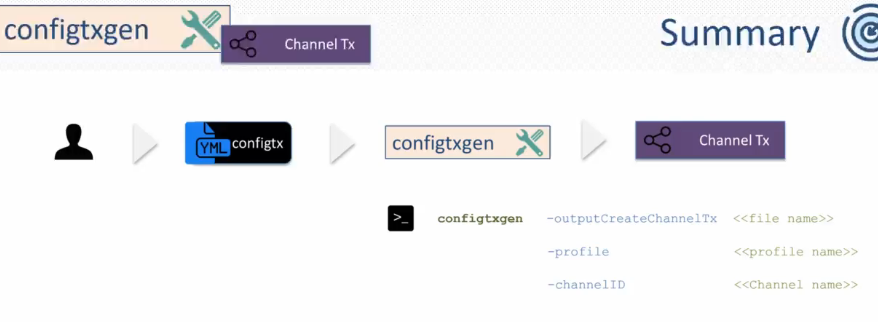
Once this has executed, you can check out the file that will be created under the temp folder.

And this is the file that has the content of the file in Json format.

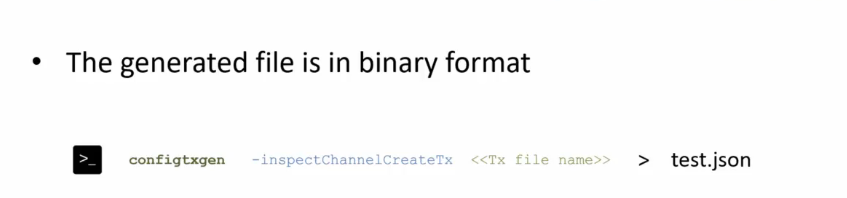


**SUMMARY**

1. showed how you can create the channel creation transaction file.
2. The developer has to set up the configuration element for the channel creation transaction in a profile in the configtxyaml file.
3. This file along with the configtxgen -output createchanneltx command will generate the channel transaction file.



1. This channel transaction file is in binary format
2. to dump out the content in Json format, you needto use the command inspect channelcreatetx and you can pipe the output of this command to a file.

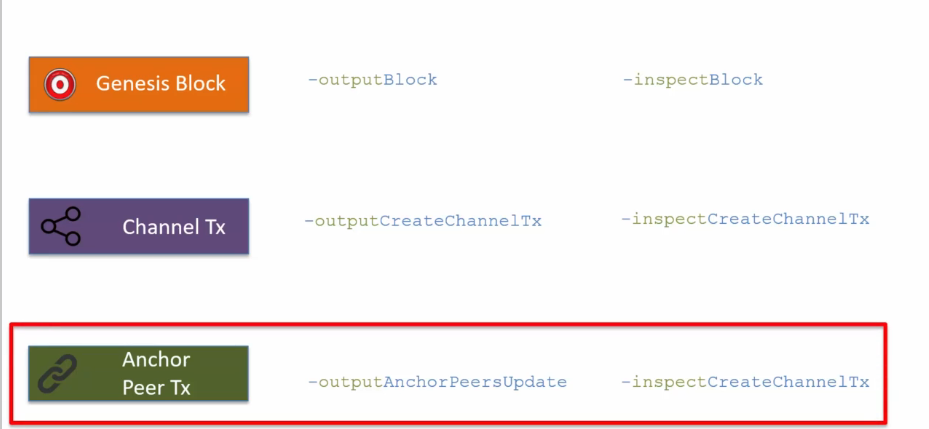


**Generating the Anchor Peer Update Transaction**

There are two objectives.

1. learn how to create the anchor Peer update transaction
2. go over the print or command that is available on the configtxgen tool.

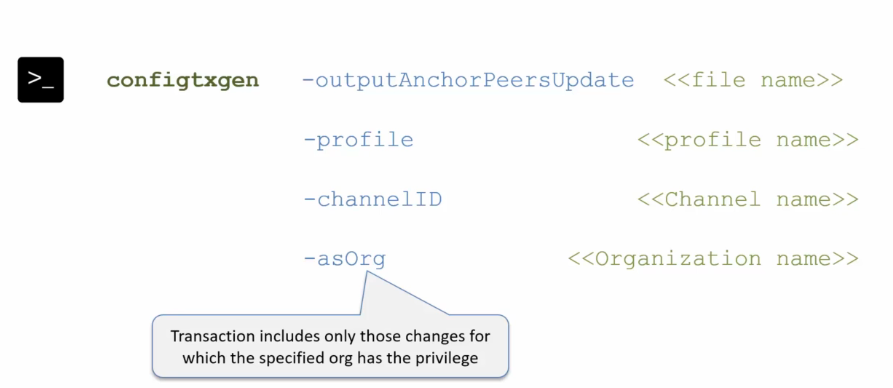
In this heading you will learn the command for creating the anchor peers update transaction.



To create the transaction to update the anchor peer, you need to use the command:



You also have to provide the profile, the channel ID, and then you also need to provide another flag as org and you have to provide the organization name.



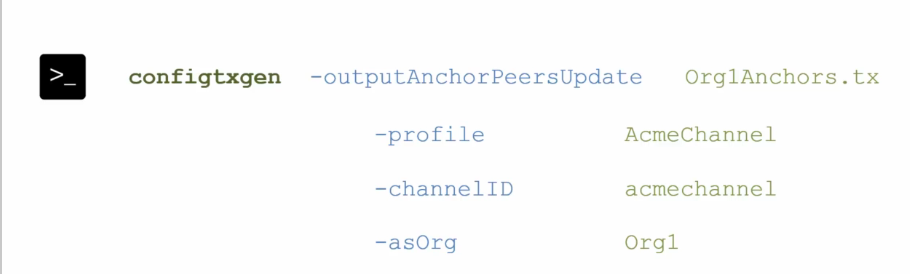
The transaction that is generated by way of this command will include only those changes for which the specified org has the privileges.

So in the case of org 1 only those anchor peer updates will be included in the transaction which are defined for org one.

So the command for updating the anchor peers for org one will look like this



This is the file name dash profile which is acme channel dash channel id name is acme channel and then as org will be org1.

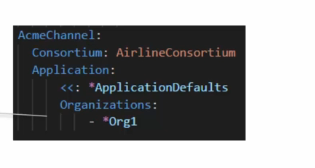


One successful execution of this command will lead to the generation of **org1anchors.txt** file in the current folder.

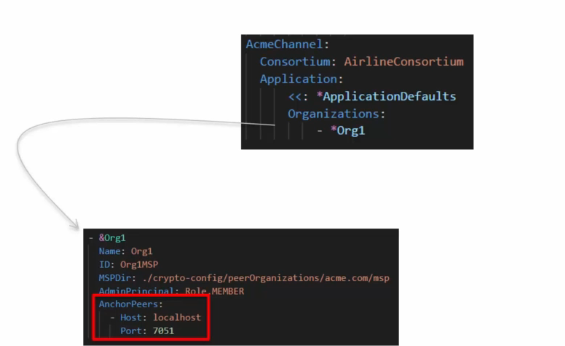
The configtxgen tool picks up the anchor peer information from the configtx yaml file using :



So in this case Acme channel which is defined here, has a set of organizations which happens to be only one organization and org one is what is specified here with the as org flag.



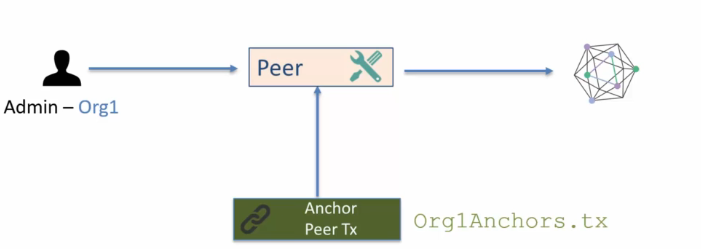
And so configtxgen picks up the information about org one and reads the subsection on anchor piers.



So this is the information which the configtxgen tool uses for generating the anchor peers update transaction.

The generated transaction file (anchor peer update Tx file)is used by the peer binary for submitting the transaction.

The way it will work in the demo scenario is that the admin of org one will use the **org1anchors.tx** file with the pier binary to submit the transaction for updating the anchor piers for org one.

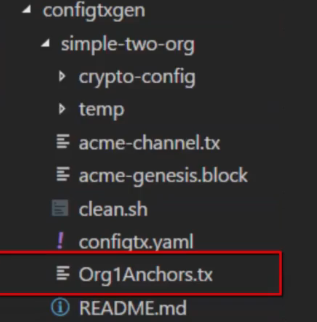


Let's go ahead and run the command

**Command to run on the terminal:**

configtxgen -outputAnchorPeersUpdate Org1Anchors.tx -profile AcmeChannel -channelID acmechannel -asOrg Org1

Once this command has executed, you will see that there is a new file that's created under the current directory.



To view the content of this binary file, you will need to use the



Here the green one is the file name

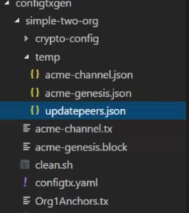
There is no special command for inspecting the update anchor peer transaction.

Let's try out the inspect command

**Command to run on the terminal:**

configtxgen -inspectChannelCreateTx Org1Anchors.tx >temp/updatepeers.json

Once you execute this command, you will see that a file has been created under the temp folder file is update piers dot Json.



The last command that I'll cover for the configtxgen tool is the command for printing the information on the organization.



* You can use the configtxgen -printorg to print the information on the organization and
* you can use the generated Json for adding new organizations.
* So you can edit the Json and create and add new organizations to the network.

The command looks like this

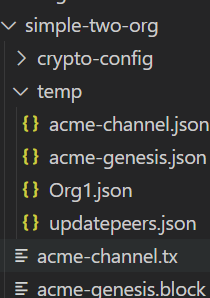


And you can pipe the Json information to a file.

Let's try the **command**:

configtxgen -printOrg Org1 > temp/Org1.json

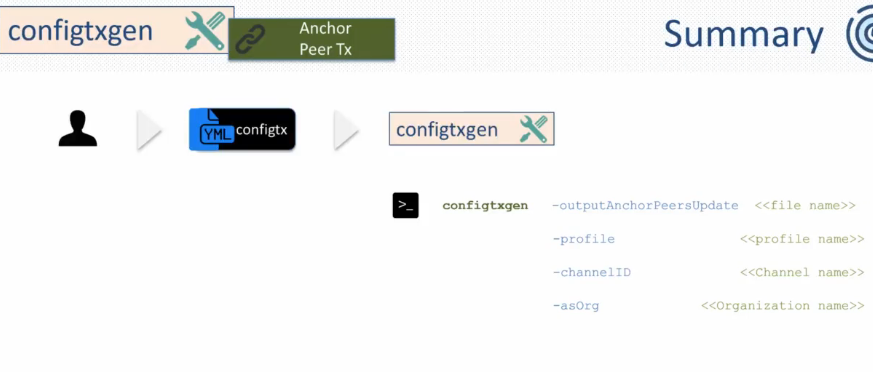
Once this command has executed there will be a new file and then you can check out the content of this file to see how the organization is set up.



**SUMMARY**

Learned:

1. how to create the Anchor Peer update transaction.
2. As a developer or as an administrator, you need to update the config file with the anchor peers information
3. and then you need to feed that file as an input to configtxgen with the command -outputanchorpeerupdate
4. you have to provide the profile and the channel ID and an additional flag need to be provided as org as org.
5. You have to specify the organization for which the anchor pier update transaction will be created.



1. inspect the generated file, you need to use the command:

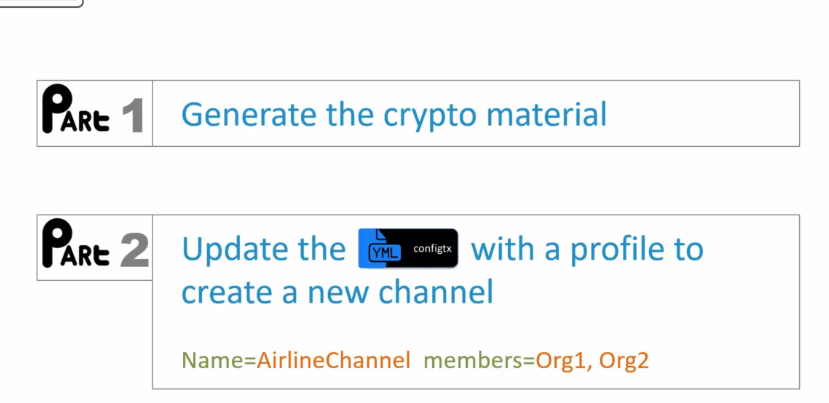


1. I also talked about a command that you can use for printing out the information on the organization That command is:



1. You provided the org name and it will read the config yaml file to pull the information about the organization.It will also merge it with the certificates for the organization.

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

****