On-Demand Training: Extract API Introduction Transcript

Hello everybody. Welcome to the introductory video on Tableau's Extract API. The Extract API allows you to programmatically create Tableau data extracts, or .tdes. The process to create TDEs in Tableau Desktop is quite easy, so why would you want to programmatically create them? Well there are a few scenarios. In some integration-minded scenarios you might need an application to create a TDE without manual intervention. Another common scenario is a third-party data tool that wants to have a TDE as an output option. Finally, the Extract API can be used to connect to data sources that Tableau does not yet connect to natively.

The Extract API can be downloaded as a library for C, C++, or Java, or as a Python module. In this video we will do two main things. First we will download and install all of the components required to use the Python version of the API. Then we will build a simple Python script to create a basic TDE.

If you're planning on using a version of the API other than Python this video should still be useful because it will explain the basic ideas of creating a TDE with the Extract API.

The first thing you should do if you're following along, is on this page that you've already navigated to, there should be a link down at the bottom - Companion Resources. This is a ZIP file that you can download and extract. It will contain three files, two of them are Python files. This first one, TrivialExample, is what we will create today to create a basic Tableau data extract. The other two files are for the video after this one, to create a TDE from a CSV file. The next thing you should do is navigate to python.org/downloads and download Python 2.7.9. I already have that downloaded so we'll start from there. Please also note that Python 3.x is not supported with the Tableau Data Extract so be sure to have Python 2.x installed. Now that we have Python 2.7 downloaded let's go ahead and install it. In almost all cases the default installation configuration should be ok so I'll just hit next through all of that.

Once that's done the next step is to make Python available to us in the command line in any directory and we'll do that by adding it to the path variable. Click the Start button, go to Computer, right-click, and choose Properties. In the left-hand column click Advanced System Settings. At the very bottom click on Environment Variables. In the bottom window there is a variable called path, highlight it and click Edit and then navigate all the way to the right side. There should be a variety of directories inside of this variable value. We'll need to add our Python directory, so if you've installed it in the same place I installed it, C:\python27 then that's what you need to add here. First make sure that this semi-colon is at the end of Python or is at the end of the system variable and then type C:\python27 and just for future uses you can go ahead and put a semi-colon at the end of that. Then go ahead and hit OK. Hit OK again and hit OK a third time. Let's go ahead and make sure that all of that has installed correctly. Click on the Start menu, type cmd and then choose that program. If you've installed Python 2.7.9 and you have added it to your path variable then you should be able to just type python and hit enter and then you should see a screen similar to this. If you get an error that Windows does not recognize that command then you've either missed one of the first two steps.

Now one key thing to note here is the version of Python, whether it's 32-bit or 64-bit. You must have that matched to

the Extract API version that you downloaded. So in this case I've downloaded and installed the 32-bit version and I'll be sure to download that same version of the Extract API. So Python's up and running, I'll go ahead and hit exit and the next step is to download and install the Extract API. Open up your browser again and navigate to tableau.com/data-extract-api. You should be able to click Download the Extract API and click Download Now and then you'll be presented with a screen that looks like this. Since we are going to download the 32-bit version of Python we'll click on this upper left-hand download. Feel free to hit pause while that downloads. I already have it downloaded and it's on my desktop.

Right-click on that file when you have it downloaded and choose Extract All. You can pretty much choose exactly where you download this to for now but I like to put it in a folder in C:/Python27 and then create a new folder called Modules, and then put it inside of Modules. So I'm downloading to C:\Python27\Modules. There's the finished folder extracted to that directory. Now we need to actually install that with Python so we have to navigate to that folder. First hit cd [space] \ and hit enter which brings us to the C drive. Then hit cd [space] and then you can type a few letters of the word python and hit the tab button to fill that in as C:\Python27. Hit another backslash because we want to go to Modules, type a few letters and hit tab, now we're in Modules. The last one is data extract with a bunch of extra words so I'll hit a few letters of that and hit tab and that's the name of the file. If you hit dir you can see that there's a file called setup.py and that's what we need to use to install this module into Python.

To do that you just type python setup.py install and hit enter. Hopefully you don't see any errors. One way to check that everything is going smoothly is type python once again to open up the python interpreter and this will allow us to just test by going import dataextract which is the name of this module. If you don't get an error, that is if you just see the next line then this worked successfully and the Python is installed as well as the Extract API Module. So I'll go ahead and hit exit and then we'll get started on actually building our script.

We'll do all of our coding for this video in a program called idle which actually installs as part of the Python package. You should be able to click Start, type the word idle, and open up that application. It starts as the Python interpreter but you can create your own scripts in here as well. To do that just hit Control+N for File New and now we have a new window to start creating our script in.

Now to start with I'll just go ahead and make sure this saves to where I want it to be. So I'll go File Save As. This is in my Extract API folder where I extracted all of the files that I downloaded at the beginning, the one with the finished examples. I'll call it TrivialExample.py. Let's get started in starting to build this Python script out.

So the first step you do in most Python scripts is to import the modules you're going to use. Now we just finished installing the Extract API module which has this name data extract. Now that's a long name and we're going to

have to use that so what I like to do is say as tde to give it an alias. We'll also need the os module for modifying files because we'll be creating files here. The first step in most Extract API files is to create the extract file. Now if you have a file and you've run it a few times the question is what do you do with that file if the .tde already exists. There's two common scenarios, one would be to do the equivalent of a full refresh. The way we do that is if the files exists then we simply delete it and then create a new one. The first thing we'll do is check to see if that file already exists. What we'll eventually call our TDE is TrivialExample.tde. We'll check if that already exists in case this is the second or nth time that we've run this script. If it does exist as I said then we'll just delete it. Which is os.remove ('TrivialExample.tde'). So now whether it existed or not there should not be that file anymore so we can go ahead and create that file. We'll create a variable called tdefile to be able to manipulate that file later in this script.

Now this line is how you actually create the extract, at this point it's a completely empty extract, but it's tde.extract or if you weren't using an alias data.Extract to call the [unsure of word - 12:25] for this extract file. This will essentially just create the file, name it a .tde and give it some other properties. Once you've created that empty file the next step is to create what's known as the table definition. This is essentially like the metadata, you're telling the extract what columns and what data types exist in this extract.

We need a variable to do everything with this table definition, we'll call it tabledef, and use the module tde and call this [unsure of word - 13:12] TableDefinition. It's an empty table definition to start with so we need to start adding some columns. tableDef.addColumn. We're going to do two simple ones, one is a rowID and one will be just a value. So one dimension and one measure. You need to define for each of these columns what type it is. In the next video we'll do an example where we'll go through a lot of these different types. In this case we'll just do a string which is the rowID. The next one will be a value which is a double or a floating point integer.

Once we've created the table definition we need to create a new table because this extract essentially has no tables at this point and we want to create that table essentially in the vision or the image of the tableDef that we just defined. You do this by creating a new variable table = tdefile, the name of the file that we create above, .addTable. The first argument is going to always have to be this Extract word, at least in the current version of the Extract API, and the second argument is the variable where we're storing the table definition, tableDef in this case. So we've created essentially our skeleton of our extract at this point. It has a table, that table is defined by this table definition which has two columns, and the last step would be to actually put data inside of that table.

The way you do this is you insert rows one by one. So what we'll do is we'll create a variable called newrow, and we'll loop through, every time through a new loop we'll redefine this newrow, and then once we're done we'll insert it into the table.

Now a row just like a table needs to be defined by some table definition. In this case we've already created that table definition. Then we'll create a loop. The idea here is we'll just go from 1-100 and throw those numbers into our extract. A very simple idea, you would of course never create an extract like this in real life. But the idea here is a simple example. So one by one we need to go through all of the columns we have and define them for this newrow variable. In this case we only have two columns so that should be pretty easy.

Now the way you define the data for each of these cells is by calling one of a variety of different methods here: setCharString, setDouble, setInteger etc. You're defining that cell by what's in it but you have to call the function that is appropriate for the type of data you're putting in it. Now the first row is a charstring you'll find up above. So that means we have to call setCharString. The first argument into any of these set arguments, setCharString, setDouble etc. is which column is it starting at 0. So this is the 0th column and this is the first column and then if we have more we would go from there. Basically I want to create something simple where it's just the word row [space] and then the number of row it is. In this case since we're looping through and we have that variable i, we'll just turn that into a string, add it to the string row and then throw that in. So the first row will say row 1, the second will say row 2 etc.

The next column is a double so we need to call setDouble. Again that's the first column, and all we want to do is throw in the integer that we're currently on so on the first iteration through the loop we just throw the number 1 into that row, on the 50th we'll throw the number 50. Now that we've created our row we want to insert it into the table with table.insert. After you're done with that, because wide space is significant in Python, be sure to un-indent to exit out of that loop. The last step is to close the TDE. If you're not familiar with Python by the way, these red lines where I start them with a pound symbol, those are comments, they aren't evaluated they're just ways to organise our file.

You must close your TDE file at the end of any Extract API script otherwise the extract will not be usable. So that's it for our creation of our extract. We started by creating the actual file, then we defined the table definition by telling the API which columns and datatypes we have. We added a table to the extract. And finally we looped through 1-100 and added a simple row, one-by-one into that extract and then once again we closed our TDE file. If everything is up and running we should be able to hit save, go to run and run the module. That will bring us back to the interpreter and if you are just presented with a new line as you see here then that's a good sign. If you get an error then there might be something wrong with either your installation of the Extract API or with your script itself. An easy way to debug your script is to go back to that finished example. It should be in C:\Extract API or wherever you decided to unzip that initial file. Open up TrivialExample_Finished and you'll be able to compare to your file. One thing that I didn't note, if you are having problems, everything in here should be case sensitive except for the things inside of strings. In fact this would be case sensitive as well.

Ok so Python executed our script and now we open up that place where we ran that script. So my script here TrivialExample was run, since we didn't specify a different directory it created that .tde Tableau Extract in the same

directory. Now just to test to make it work we can double-click on TrivialExample here and we're presented with the data window in Tableau and we can go to the worksheet and make sure all of the variables are there, maybe even test the columns out. So it looks like everything worked, I hope that worked for you. In the next video we'll actually connect to a CSV as an example of connecting to real data and build an extract from that using the Extract API. Hope everything went well. See you on the next video.