# Apitax

Pronounced: *ahhp-ehh-tax*

**tl;dr: Code examples are at the very *bottom* of this documentation; however, I highly suggest you read through the documentation to learn what is possible.**

Finally, as Apitax features an exponential amount of various interactions, not all of them are documented here. Experiment with the syntax, Learn from the syntax, and Enjoy Apitax!

## Builds

### Packaging Instructions

* Use either Powershell or Bash
* Building the package: python setup.py sdist bdist\_wheel
* Upload the package to Pypi: twine upload dist/\* -r pypi
* More information can be found here: https://gist.github.com/ShawnClake/759e9d09af868ef18f8c7b39d1684ad4

### Readme Conversion Instructions

* Install pandoc if it is not already installed: sudo apt-get install pandoc
* Run the command: pandoc -o readme.docx -f markdown -t docx README.md

### Compile Antlr Grammar from another directory

This is sometimes nessecary due to a bug in the antlr compiler with regards to paths \* Download the antlr compiler .jar file and save it somewhere. \* Inside of the directory where the .jar was saved, create the following folder heirarchy \* build \* src \* scripts \* logs \* Create a .bat or .sh file (dependent on OS) and add the following script:

java -jar ~/grammar/antlr-4.7.1-complete.jar -Dlanguage=Python3 ~/grammar/src/AhLex210.g4  
java -jar ~/grammar/antlr-4.7.1-complete.jar -lib ~/grammar/src -o ~/grammar/build -listener -visitor -Dlanguage=Python3 ~/grammar/src/Ah210.g4  
cp -r ~/grammar/build/\* ~/Apitax/apitax/grammar/build  
cp ~/grammar/src/AhLex210.g4 ~/Apitax/apitax/grammar/src  
cp ~/grammar/src/Ah210.g4 ~/Apitax/apitax/grammar/src

* Anytime you want to make changes to the grammar, do it from the new ~/grammar/src directory and run the script

## Documentation and Usage

### Commandtax - Data Gathering, Manipulating, Usage

#### Existing

* script <pathToSomeScript>
  + Sequences of commands to automate the execution of API requests
  + Scripts can include additional script commands which effectively means scripts can be nested
  + Scripts are run in order from top to bottom, if a nested script is found it executes the nested script before continuing through the current script
* custom <someCustomCommand>
  + Processes a custom request which is not baked into the utility
  + Parameters
    - --get : Uses a get method
    - --post : Uses a post method
    - --put : Uses a put method
    - --patch : Uses a patch method
    - --delete : Uses a delete method
    - --url : (string) The endpoint
    - --data-post : (json string) Any post data
    - --data-param : (json string) Any query parameters ie. Endpoint.com/something?this=queryparam
    - --data-path : (json string) Any url path variables ie. Endpoint.com/users/{path\_var}/show

#### Coming Soon

* shell <someCommand>
  + Runs the command in the shell and returns the response from the shell

### Scriptax - Control Flow, Scoping, and Automation

#### Existing keys

* ct("<someCommand>")
  + Commandtax execution
  + A command is executed during the parsing of the line and its response is returned
* set <someVar> = <someValue>
  + Sets a variable
  + Supports expressions, strings, numbers, booleans, dictionaries, lists, and commandtax responses
* "this is a string {{ someVar }}"
  + Injects the contents of a variable
  + Fancy stuff is possible such as: set newVar = ct("{{someVar}}")
* {{ r: someResponse }}
  + Injects the response of some request
* import ct("some commandtax")
  + Executes a command and imports the response to the current scope
* export ct("some commandtax"), export someVar
  + Imports the values to the current scope and exports them to allow a parent scope to access these values
* name <someName>
  + Sets the reference name of the script.
  + Supports strings and expressions
* log("log some output to the console & log file")
  + Supports expressions
* cast(<someVar>, type)
  + Casts a variable to a type
  + Valid types are: str, num, dict, list
* // some comment
  + Inline comment
* /\* some comment spanning multiple lines \*/
  + Block comment

#### Coming Soon

* {% if <someCondition> %} <someExecution>
* {% if <someCondition> %}: <someSetOfExecutionSpanningMultipleLines> {% endif %}
* return <someDataOrOptionallyNoData>
  + Stops the processing of the current script and gives control back to the parent script
  + The parent script, if importing this subscript, will have access to any data returned by this subscript. The returned data will be directly under the subscripts namespace in the dataStore
    - For example: {"vars": {"someSubScript":{"return": "{"this is": "the returned data"}"}}}
* <someVariable> to <type>
  + Cast a variable to a type
  + Types: boolean, string, int, float, double, json, object
  + Type checking is not preformed and is expected of the user to know what they are trying to do
* {% for <someNewVariable> to range(<someStart>,<someEnd>) %} <someExecution>
* {% for <someNewVariable> to <someEnd> %} <someExecution>
* {% for <someNewVariable> in <someExistingVariableObject> %} <someExecution>
* {% for oneOfTheCasesListedAbove %}: <someSetOfExecutionSpanningMultipleLines> {% endfor %}
* require
  + Provide a means for a parent script to gather all of the data it will need to run from the executing user

#### Tidbits

* You can use arrays via dot notation
  + set someVar.1 = num1
  + set someVar.2 = num2
  + set someVar.{{counter}} = num3
  + When doing this, the first usage of a variable must either be someVar = "{}" or an index as part of that object. Failure to do this will result in errors being thrown.

### Commandline Interface (CLI)

* You can activate Apitax from the CLI directly without needing a wrapper package
* Run the tests.py file found in the root Apitax directory, and supply the following arguments
  + --cli : (Optional) Quickly select CLI mode
  + -- web : (Optional) Quickly select web server mode
  + --debug : (Optional) Output the request response status, headers, and body
  + -u : (Optional) Specify the authentication username - Only applicable in CLI mode
  + -p : (Optional) Ask for password input right away. If -u is specified, but this is not, the application will ask for a new set of credentials for authentication. But, it will use the -u value for any username fields within further requests. This allows someone to authenticate as admin, but run commands applicable to another user.
  + -r : (Optional) The request - Only applicable in CLI mode
  + -s : (Optional) Run a script of requests specified by the file path , authentication for all of these requests are specified by -u & -p

### Modes

* CLI : Make a request from the command line
* WEB : Start the web server interface
* Grammar Test : Run a test of the parsing and grammars

### Configuration

* Please utilize the config file found in the main apitax directory
* Eventually these parameteres will be programmatically assignable
* Eventually a custom config file path will be able to be passed in

Example Configuration File:

config.txt  
  
[Config]  
auth-endpoint = "<someEndpoint>"  
driver = ApitaxTests  
port = <somePortForTheWebServer>  
ip = <someIPForTheWebServer>  
log = true  
log-file = logs/log.log  
log-colorize = true  
default-mode = cli  
default-username = <someAuthUsername>  
default-password = <someAuthPassword>

### Supported Authentication

* Authentication is prebuilt for HTTP Basic and Token based authentication
* Authentication is largely left up to a developer in custom scenarios
  + Driver files facilitate this requirement

### Drivers and Plugins

* Drivers and plugins are used to extend the functionality of Apitax to an arbitrary API
* While dynamically injecting drivers and plugins is on the todo list, for now all plugins must go into the apitax/drivers/plugins directory
* Each driver requires at least a core plugin file
  + This file describes any custom parameters with regards to authentication, endpoints, and the requirements of the specific API the driver is built for
* Optionally, a driver can also include commandtax plugins which are used to specify shortcut commands
  + To do this, a core commandtax plugin file is required as well as a commandtax plugins directory. The core commandtax plugin file must contain the suffic Commands
    - Core commandtax plugins file: apitax/drivers/plugins/commandtax/CoreFileCommands.py
    - Core commandtax plugins directory: apitax/drivers/plugins/commandtax/<theNameofTheDriver>/
  + Inside of this new plugins directory, you can create shortcut files to route specialized commands. You can see examples of this in the apitax source code with regards to the ApitaxTests driver and plugin files

### Examples of Apitax in Action:

#### Commandtax Examples

```  
custom --get --url <someEndpoint>  
custom --get --url <someEndpoint> --data-param '{"is\_domain": true}'  
custom --post --url <someEndpoint> --data-post '{"title": "im the title"}'  
custom --put --url <someEndpoint>   
custom --patch --url <someEndpoint>  
custom --delete --url <someEndpoint>   
custom --get --url <someEndpoint> --data-param '{"user.id": "1"}'  
custom --get --url <someEndpoint>/with/some/{ohyear}/url/params/{981} --data-param '{"is\_domain": true}' --data-path '{"ohyeah":"no", "981": "yes"}'  
```

script ~/path/to/my/script.ah

#### Scriptax Examples

// tristan.ah  
  
set jen = 6.9 / 69  
  
set paris = {{jen}} + 5  
  
set tristan = "runescape"  
  
log(tristan + "hi")  
  
ct("domain list all")  
  
export ct("script apitax/grammar/scripts/jen.ah")  
  
log(quinn.jordan.shawn)  
  
import ct("custom --get --url https://jsonplaceholder.typicode.com/posts")  
  
set noway = {{r:1.0.title}}

// jen.ah  
  
name "quinn"  
  
set qwer = "im in jens script"  
  
export qwer  
  
export ct("script apitax/grammar/scripts/shawn.ah")  
  
set iam.hope.this.works = ct("custom --get --url https://jsonplaceholder.typicode.com/users")  
  
export iam

// shawn.ah  
  
name "jordan"  
  
set shawn = 55  
  
export shawn  
  
set whynotme = '["first", "second", "third"]'  
  
set whyme = cast(whynotme, dict)

/\*   
A major test script used in our personal testing procedure  
test3.ah  
\*/  
  
7 + 3 \* (10 / (12 / (3 + 1) - 1))  
  
7 + 3 \* (10 / (12 / (3 + 1) - 1)) / (2 + 3) - 5 - 3 + (8)  
  
(10 / (12 / (3 + 1) - 1))  
  
(2 + 3)   
  
3\*(10 / (12 / (3 + 1) - 1))  
  
3 / 5 \* 2  
  
5 - 2.2  
  
3/5\*(10 / (12 / (3 + 1) - 1))  
  
7 + (((3 + 2)))  
  
5 \* 6 \* 3 \* 2  
  
36 / 6 / 2 / 3  
  
5 + 2 + 3 + 4  
  
5 - 2 - 3 - 5  
  
name "bobbyjoe"  
  
name "keystones"  
  
set shawn = 65  
  
set derp = "thederpiest"  
  
name derp  
  
{{simble.jim}}  
  
{{simble.lionking.shawn}}  
  
{{r:simble.lionking.1.role\_assignments.0.user.id}}  
  
{{bobby.projects.0.is\_domain}}  
  
set mynum = 3  
  
1 + 1 + {{mynum}}  
  
name derp  
  
set jimmy = {{mynum}} + 3  
  
set somestring = "heck no: " + {{derp}} + " :( " + derp  
  
set come.on = "noway"  
  
set come.here = dict('{"myname": "is not cool", "youknow" : "?"}')  
  
log({{derp}} + " " + somestring + " ===> " + cast(come, str))  
  
set ksjhdg = '{"myname": "is not cool", "youknow" : "?"}'  
  
set something = cast(ksjhdg, dict)  
  
set heck.no = cast(mynum, dict)  
  
set cough = '["test0", "test1", "test2"]'  
  
set heck.yes = cast(cough, dict)  
  
{{heck.yes.1}}  
  
set me = "blah blah blah {{heck.yes.1}}"  
  
export me  
  
//import ct("script apitax/grammar/scripts/test2.ah")  
  
//set amanda = {{test2apitaxtest.test4apitaxtest.nooby}} + " or maybe jk"  
  
//log(amanda)