Template Definition Language - TDL

https://campus.barracuda.com/doc/96026437/

The TDL is a human read/writable language for describing Configuration Templates. TDL files can be translated into a JSON definition file using `tdltool`, which is deployed on every CC with firmware version 8.2 or higher.

The TDL is composed of several sections, which all follow the same syntax:

```
[<sectionname> <attribute>=<value> ...]
<parameter> <attribute>=<value> ...
...
```

Character Sets

The section names, parameter names, and attributes are simple strings without " or ' and are defined as

```
[a-zA-Z0-9]+
```

Values can be more complex strings and can have " or ' around them and are defined as

```
\"[a-zA-Z0-9\'\(\)\ \-\_\.\,\/\*\+\=\#]+\" or \'[a-zA-Z0-9\"\(\)\ \-
\_\.\,\/\*\+\=\#]+\'
```

Sections

The TDL provides a set of keywords to separate script code into logical sections. Every section provides a subset of further keywords as attributes.

The following sections are allowed and recognized:

template

The template section defines the basic parameter of the ConfTemplate.

Attribute	Required	Use	Example

name	Y	Name/ID of the template	[template name="mytemplate"]
label		The label can be displayed in a GUI	[template name="mytemplate" label="my sweet template"]
description		Description to be displayed in a GUI	[template name="mytemplate" label="my sweet template" description="Very long descriptive text."]

The template section can be attributed by a parameter section. (For an explanation of the parameter section, see the following paragraph).

Parameter	Attribute	Required	Usage	Example
meta	version		An optional version of the TDL language to allow explicit use of a different version than the current default.	meta version='1.0'

parameters

The section `parameter` does not have any attributes.

Parameter	Attribute	Required	Usage	Example
<unique parameter name>			The template input parameters have to be unique and can be referenced in other parts of the template	
	type	Y	Parameter type	managementIp type=ipv4
	default		A default value for the parameter. For types, int-only numbers without " or ' are allowed, for bool only true and false are allowed values. All other values have to have " or ' around them.	password type=securestring default="a"
	min		An integer value, which defines for integers the minimal allowed value, for strings the minimal allowed length.	
	max		An integer value, which defines for integers the maximal allowed value, for string the maximal allowed length.	test type=int default=9 min=1 max=10
	label		A label to be displayed in a GUI.	
	description		A description that is to be displayed in a GUI.	

types

These data types serve as an intrinsic input filter for certain numbers and (special) characters that a user enters at runtime and ensures that only valid data hits the processing level.

Data Type	Example	Character Set, allowed Values	Note
alphanum	AlphanumericWithNumber1	A..Z, a..z, 0..9	Note that the example in the second column is a string. Enter any text with the allowed characters without quotation marks.
alphanumdash	Alphanumeric Text With Dash And Number-1	A..Z, a..z, 0..9	Note that the example in the second column is a string. Enter any text with the allowed characters without quotation marks.
alphanumspacedash	Alphanumeric Text With Dash And Number - 1	A..Z, a..z, ' ', 0..9	Note that the example in the second column is a string. Enter any text with the allowed characters without quotation marks.
bool		true, false	The Boolean value is set by a check box.
boolean			
dynamiclist	p1		A dynamic list of entries (e.g., visible in a menu list) that depends on the actual parameter.
email	support@barracuda.com		An example of an email address.
int	1		A number without a decimal point.
ip	18.66.15.47	Each number: 0..255	An IP address consists of a sequence of 4 numbers, separated by a period. The range of each number must be in the range from 0 to 255.
ipmask	255.255.255.0	Each number: 0..255	An IP mask consists of a sequence of 4 numbers, separated by a period. The range of each number must be in the range from 0 to 255. This type of IP number is used for applying special bitwise arithmetic operations by 'masking' certain bits in a general IP address.
ipnet	18.66.15.47/24	The highest number for the appendix: 32 (= 4+8bits)	This type of IP address is used for describing networks and is followed by a slash and a number between 0 and 255. This appendix signifies that certain bitwise masking operations are implicitly done when this type of IP network address is used.
ipv4	18.66.15.47	Each number: 0..255	An IP address that consists of a sequence of 4 numbers, separated by a period. The value of each number must be in the range from 0 to 255.

ipv4cidr	/16		CIDR stands for 'Classless Inter-Domain Routing' and is used as an appendix when using IP network addresses. The number behind the slash signifies the number of bits to break up an IP address into two parts where the first part is then interpreted as the 'Network' part and the second part is used to identify a certain number of hosts in that network. An IPv4 address consists of 4 bytes where each byte is represented by 8 bits, making a total of 32 bits. As an example, the bitwise representation of the IPv4 address 213.86.117.16 would be 11010101.01010110.01110101.00010000 . The number behind the slash tells how many bits with the value of '1' are to be used as a mask counting from the very left in the binary representation to the right, e.g. /16 resulting in '11111111.11111111.00000000.00000000'. By arithmetically applying this number bitwise with the logical AND operator to the binary network representation of the IPv4 address 213.86.117.16, the result is 11010101.01010110.00000000.00000000., which is in decimal numbers 213.86.0.0.
ipv6ipornet			
ipv4mask	255.255.0.0	Each number: 0..255	An ipv4mask is an IP number to be used as a mask for arithmetical bitwise operations to calculate network addresses.
ipv4net	18.66.15.47/24	The highest number for the appendix: 32 (= 4+8bits)	an ipv4net is the number to identify an IP network based on an IP number followed by an ipv4cidr.
ipv6	::1		An IP address that consists of a sequence of 8 numbers, separated by a period. The value of each number must be in the range from 0 to 255.
ipv6mask	ffff:ffff:ffff:ffff:0000:0000:0000:0000	Each number: 0..ffff	An IPv6 mask consists of a sequence of 8 numbers, separated by a period. Each number must be in hexadecimal format. The range of each number must be in the range from 0 to ffff. This type of IP number is used for applying special bitwise arithmetic operations by 'masking' certain bits in a general IP address.
ipv6net	2022:db8:abcd:0012::0/64	Each number: 0..ffff	The ipv6net is the number to identify an IPv6 network based on an IP number followed by a CIDR.
ipv6prefixlength			
mac	00:01:f3:34:44:2f	Each number: 0..ff	A MAC (Media Access Control) address is a sequence of 6 bytes forming a hardware address to identify interfaces on a network. Each number must be in hexadecimal format.
majorminorversion	9.0		A number naming the version for a software/firmware release. The number preceding the decimal point indicates the major version number, and the number following next to the decimal point indicates the minor version number.
number	12, 1463, 38422, ...		A sequence of numeric characters.

securestring	`passwd1`	All characters of the ASCII character set.	
securestringhash		All characters of the ASCII character set.	
string	'Text from ASCII Characters'	All characters of the ASCII character set.	Note that the example in the second column is a string. Enter any text with the allowed characters without quotation marks.

variables

The section `variables` does not have any attributes.

Parameter	Attribute	Required	Usage	Example
<unique variable name>			The template variables can contain expressions to derive values from parameters to be used as ConfUnit inputs.	
	type	Y	Variable type.	
	expression		A formula that can use parameters and other variables to calculate a new value.	hospitalIP type=ipv4 expression="ipadd(ipgetaddr(hospitalNetwork), hospitalIPOffset)"
	value		A variable can have a static value.	isenabled type=bool value=false

confunit

Attribute	Required	Usage	Example
type	Y	One of the ConfUnit types supported on the CC	
name	Y	The name of the ConfUnit instance. This name has to be unique in the ConfTemplate.	[confunit type=sccore name=mycore]
condition		A confunit can be defined conditionally in a ConfTemplate. This can reference a bool parameter or variable from the parameters or variables section.	[confunit type=sccore name=mycore condition=isenabled]

The unit `confunit` accepts the following parameters:

Parameter	Attribute	Required	Usage	Example
<parameter name as expected by the confunit>		Depends on the confunit definition		
	value		int and bool values have to be given without " or ', all other types have to have them.	location value="Vienna"
	paramref		Reference to a parameter, which has to be defined in the parameters section.	boxPasswordHash paramref=password
	variableref		Reference to a variable, which has to be defined in the variables section.	

Structuring Keywords

Objects

Objects are structure-like composed parameters. If a Configuration Unit requires an object as input, this can be expressed by hierarchical parameters, where hierarchies are separated by a '.'. E.g. if an object is called a basket with members avocado and pineapple, which are both integers, this can be expressed using the following syntax:

- `basket.avocado value=5`
- `basket.pineapple value=2`

Arrays

Array values can be expressed by adding an index in brackets [] to the parameter name. The value itself can be an explicit value or a reference. Missing index values will be augmented with empty entries, as indices have to be in a linear sequence.

Example

```
[template name="mytemplate" label="my sweet template"]

[parameters]
lan type=string default="42.0.0.1/24" label="My favorite LAN"
description="This LAN is so important."
password type=securestring default="a"
switch type=bool default=true
mynumber type=int default=4 label="My number"
mystring type=string label="A string" description="There is no default."

[confunit type=sccore name=mycore]
description value="test"
location value="Vienna"
dataNet value="data1_clustr_1"
dnsServerIps[0] value="8.8.8.8"
dnsServerIps[1] value="9.9.9.9"
dnsServerIps[2] paramref=lan
boxPasswordHash paramref=password
sshAccess value=true
enableNtp value=true
linkselection.enable value=true
primaryInterface value="Wan"
linkselection.backupInterface value="Wwan" number value=9

[confunit name=myvpn type=scvpn]
tunnel.mode value="TCP"
encryption value="AES256"

[confunit name=mylan type=sclan]
mode value="manual"
lanPort value="Lan1"
net paramref=lan
networkMapping value=false
```

Converting between ConfTemplate and JSON

ConfTemplates include the same information as JSON files, but unlike JSON files ConfTemplate files are better human-readable. The equivalent TDL format can be translated into JSON format and back without information loss. You can either use the ConfTemplate Manager in FW Admin to do this or use the command line tool "`tdltool`".

TDL CLI Tool

`tdltool` provides the following options:

Usage:

```
-t|--tdl <TDL file> to be parsed, output is JSON.
-j|--json <JSON file> to be parsed, output is TDL.
-r|--range <range> range on CC for retrieving and deploying.
-c|--cluster <cluster> cluster on CC for retrieving and deploying.
-n|--name <name> name of template for retrieving, output is TDL.
-i|--deploy Install or update the template read from a file to a range or
cluster.
-o|--output <output file>.
-a|--docall ConfUnit documentation of all ConfUnits.
-l|--list List available ConfUnits.
-d|--doc <Conf Unit Type> ConfUnit documentation.
-v|--verbosity <n>
-h|--help
```

Example

Convert a TDL file into a JSON file:

```
  tdltool -t newtemplate.tdl -o newtemplate.json
```

Convert a JSON file into a TDL file:

```
  tdltool -j newtemplate.json -o newtemplate.tdl
```

	TDL File Content	JSON File Content
Conversion command	tdltool -t newtemplate.tdl -o newtemplate.json	tdltool -j newtemplate.json -o newtemplate.tdl

```
[template name='demo' label='A ConfTemplate']
meta product='CGF'

[parameters]
mypassword type=securestring
mip type=ipv4

[confunit type=core name='myCgCore']
description value='better demo description'
managementNetwork.defaultGateway value='192.168.0.1'
managementNetwork.interface value='p1'
managementNetwork.ip paramref=mip
managementNetwork.mask value='255.255.255.0'
managementNetwork.sharedIps[0].ipAddress paramref=mip
password paramref=mypassword

[confunit type=firewall name='firewall']

[confunit type=sharedNetwork name='sharedNetwork']
defaultGateway value='11.22.33.1'
interface value='p2'
networkAddress value='11.22.33.0/24'
sharedIps[0].alias value='second'
sharedIps[0].ipAddress value='11.22.33.44'
trustLevel value='trusted'

[confunit type=dhcp name='dhcp']
dnsServerIp value='9.9.9.9'
endIp value='22.33.44.100'
startIp value='22.33.44.10'
subnet value='22.33.44.0/24'

[confunit type=dns name='dns']
forwarders[0] value='8.8.8.8'
forwarders[1] value='9.9.9.9'
```

```
{
 "name": "tutorial",
 "displayName": "A ConfTemplate",
 "meta": {
  "product": "CGF"
 },
 "parameters": [
  {
   "name": "mip",
   "type": "ipv4"
  },
  {
   "name": "mypassword",
   "type": "securestring"
  }
 ],
 "confUnits": [
  {
   "name": "myCgCore",
   "type": "core",
   "properties": [
    {
     "name": "description",
     "value": "better demo description"
    },
    {
     "name": "managementNetwork",
     "object": {
      "properties": [
       {
        "name": "defaultGateway",
        "value": "192.168.0.1"
       },
       {
        "name": "interface",
        "value": "p1"
       },
       {
        "name": "ip",
        "parameterRef": "mip"
       },
       {
        "name": "mask",
        "value": "255.255.255.0"
       },
       {
        "name": "sharedIps",
        "array": {
         "elements": [
          {
           "object": {
            "properties": [
             {
              "name": "ipAddress",
              "parameterRef": "mip"
             }
            ]
           }
          }
         ]
        }
       }
      ]
     }
    },
    {
     "name": "password",
     "parameterRef": "mypassword"
    }
   ]
  },
  {
   "name": "firewall",
   "type": "firewall"
  },
  {
   "name": "sharedNetwork",
   "type": "sharedNetwork",
   "properties": [
    {
     "name": "defaultGateway",
     "value": "11.22.33.1"
    },
    {
     "name": "interface",
     "value": "p2"
    },
    {
     "name": "networkAddress",
     "value": "11.22.33.0/24"
    },
    {
     "name": "sharedIps",
     "array": {
      "elements": [
       {
        "object": {
         "properties": [
          {
           "name": "alias",
           "value": "second"
          },
          {
           "name": "ipAddress",
           "value": "11.22.33.44"
          }
         ]
        }
       }
      ]
     }
    },
    {
     "name": "trustLevel",
     "value": "trusted"
    }
   ]
  },
  {
   "name": "dhcp",
   "type": "dhcp",
   "properties": [
    {
     "name": "dnsServerIp",
     "value": "9.9.9.9"
    },
    {
     "name": "endIp",
     "value": "22.33.44.100"
    },
    {
     "name": "startIp",
     "value": "22.33.44.10"
    },
    {
     "name": "subnet",
     "value": "22.33.44.0/24"
    }
   ]
  },
  {
   "name": "dns",
   "type": "dns",
   "properties": [
    {
     "name": "forwarders",
     "array": {
      "elements": [
       {
        "value": "8.8.8.8"
       },
       {
        "value": "9.9.9.9"
       }
      ]
     }
    }
   ]
  }
 ]
}
```