gns3-converter Documentation

Release 1.2.4

Daniel Lintott

December 20, 2015

Contents

GNS3 Converter is designed to convert old ini-style GNS3 topologies (<=0.8.7) to the newer version v1+ JSON format for use in GNS3 v1+

The converter will convert all IOS, Cloud and VirtualBox devices to the new format. It will also convert all QEMU based devices (QEMU VM, ASA, PIX, JUNOS & IDS). VPCS nodes will be converted to cloud devices due to lack of information the 0.8.7 topology files.

For topologies containing snapshots, the snapshots will also be converted to the new format automatically.

Contents:

CHAPTER 1

Installation

1.1 Linux

1.1.1 Requirements

- Python 3.3+
- ConfigObj

1.1.2 Instructions

On linux gns3-converter can be installed using pip. Simply type:

```
pip install gns3-converter
```

or easy_install:

```
easy_install gns3-converter
```

alternatively you can manually install gns3-converter, by downloading the source from http://pypi.python.org/pypi/gns3-converter (you'll need to also install ConfigObj):

```
python setup.py install
```

1.2 Windows

1.2.1 Instructions

On windows you can install gns3-converter using the installer provided at: https://github.com/dlintott/gns3-converter/releases

CHAPTER 2

Using gns3-converter

Convert old ini-style GNS3 topologies (<=0.8.7) to the newer version 1+ JSON format

```
usage: gns3-converter [-h] [--version] [-n NAME] [-o OUTPUT] [--debug] [-q]
                      [topology]
```

Positional arguments:

topology	GNS3 .net topology file (default: topology.net)

Options:

--version	show program's version number and exit
-n, --name	Topology name (default uses the name of the old project directory)
-o, --output	Output directory
--debug=False	Enable debugging output
-q=False, --quiet=False	Quiet-mode (no output to console)

2.1 Example

By default the converted topology will be output to the current working directory.

To convert a topology from the folder containing the topology.net file just type:

```
gns3-converter
```

Alternatively you can specify a topology file to convert on the command line:

```
gns3-converter ~/GNS3/Projects/CCNA_1/topology.net
```

If the relevant configs are also present alongside the topology file these will be copied to the new topology and renamed accordingly.

If you wish to output the converted topology to a different destination this can be done using the -o or –output argument like this:

```
gns3-converter -o ../output
```

or

```
gns3-converter --output ../output
```

The name of the converted topology is taken from the folder containing the topology file. For example a topology in ~/GNS3/Projects/CCNA_1/topology.net will be named CCNA_1.

It is also possible to specify a name for the new topology using the -n or –name in the same way as specifying the output directory.

CHAPTER 3

gns3converter modules

3.1 gns3converter.adapters

Convenience module for adapters containing:

- Adapter and port number/type matrix
- Port type conversions (short to long)

3.2 gns3converter.converter

This class is the main gns3-converter class

class gns3converter.converter.**Converter**(*topology*, *debug=False*)
Bases: object

GNS3 Topology Converter Class

Parameters

- **topology** (*str*) – Filename of the ini-style topology
- **debug** (*bool*) – enable debugging (Default: False)

add_node_connection(*link*, *nodes*)
Add a connection to a node

Parameters

- **link** (*dict*) – link definition
- **nodes** (*list*) – list of nodes from *generate_nodes()*

static convert_destination_to_id(*destination_node*, *destination_port*, *nodes*)
Convert a destination to device and port ID

Parameters

- **destination_node** (*str*) – Destination node name
- **destination_port** (*str*) – Destination port name
- **nodes** (*list*) – list of nodes from *generate_nodes()*

Returns dict containing device ID, device name and port ID

Return type dict

static device_id_from_name (*device_name*, *nodes*)
Get the device ID when given a device name

Parameters

- **device_name** (*str*) – device name
- **nodes** (*list*) – list of nodes from *generate_nodes()*

Returns device ID

Return type int

generate_images (*pixmaps*)
Generate the images list and store the images to copy

Parameters **pixmaps** (*dict*) – A dict of converted pixmaps from the old topology

Returns A list of images

Return type list

generate_links (*nodes*)
Generate a list of links

Parameters **nodes** (*list*) – A list of nodes from *generate_nodes()*

Returns list of links

Return type list

generate_nodes (*topology*)
Generate a list of nodes for the new topology

Parameters **topology** (*dict*) – processed topology from *process_topology()*

Returns a list of dicts on nodes

Return type list

static generate_notes (*notes*)
Generate the notes list

Parameters **notes** (*dict*) – A dict of converted notes from the old topology

Returns List of notes for the the topology

Return type list

static generate_shapes (*shapes*)
Generate the shapes for the topology

Parameters **shapes** (*dict*) – A dict of converted shapes from the old topology

Returns dict containing two lists (ellipse, rectangle)

Return type dict

static get_node_name_from_id (*node_id*, *nodes*)
Get the name of a node when given the node_id

Parameters

- **node_id** (*int*) – The ID of a node
- **nodes** (*list*) – list of nodes from *generate_nodes()*

Returns node name

Return type str

static get_port_name_from_id(*node_id*, *port_id*, *nodes*)
Get the name of a port for a given node and port ID

Parameters

- **node_id** (*int*) – node ID
- **port_id** (*int*) – port ID
- **nodes** (*list*) – list of nodes from *generate_nodes()*

Returns port name

Return type str

static get_sections(*config*)
Get a list of Hypervisor instances

Parameters **config** (*ConfigObj*) – Configuration from *read_topology()*

Returns configuration sections

Return type list

static port_id_from_name(*port_name*, *device_id*, *nodes*)
Get the port ID when given a port name

Parameters

- **port_name** (*str*) – port name
- **device_id** (*str*) – device ID
- **nodes** (*list*) – list of nodes from *generate_nodes()*

Returns port ID

Return type int

process_topology(*old_top*)
Processes the sections returned by get_instances

Parameters **old_top** (*ConfigObj*) – old topology as processed by *read_topology()*

Returns tuple of dicts containing hypervisors, devices and artwork

Return type tuple

read_topology()
Read the ini-style topology file using ConfigObj

Return config Topology parsed by `ConfigObj`

Return type ConfigObj

topology
Return the topology filename the converter is working on

Returns topology filename

Return type str

3.3 gns3converter.interfaces

Anything to do with interfaces, also contains:

- INTERFACE_RE for matching interfaces in a .net topology
- ETHSWINT_RE for matching Ethernet switch port in a .net topology

class gns3converter.interfaces.**Interfaces**(*port_id*)
Bases: object

Base Interface Class

Parameters **port_id** (*int*) – starting port ID

3.4 gns3converter.main

gns3converter.main.**copy_configs**(*configs*, *source*, *target*)
Copy dynamips configs to converted topology

Parameters

- **configs** – Configs to copy
- **source** (*str*) – Source topology directory
- **target** (*str*) – Target topology files directory

Returns True when a config cannot be found, otherwise false

Return type bool

gns3converter.main.**copy_images**(*images*, *source*, *target*)
Copy images to converted topology

Parameters

- **images** – Images to copy
- **source** – Old Topology Directory
- **target** – Target topology files directory

Returns True when an image cannot be found, otherwise false

Return type bool

gns3converter.main.**copy_instructions**(*source_project*, *dest_project*)

gns3converter.main.**copy_topology_image**(*source*, *target*)
Copy any images of the topology to the converted topology

Parameters

- **source** (*str*) – Source topology directory
- **target** (*str*) – Target Directory

gns3converter.main.**copy_vpcs_configs**(*source*, *target*)
Copy any VPCS configs to the converted topology

Parameters

- **source** (*str*) – Source topology directory

- **target** (*str*) – Target topology files directory

gns3converter.main.**do_conversion**(*topology_def*, *topology_name*, *output_dir=None*, *debug=False*, *quiet=False*)

Convert the topology

Parameters

- **topology_def** (*dict*) – Dict containing topology file and snapshot bool. For example: `{'file': filename, 'snapshot': False}`
- **topology_name** (*str*) – The name of the topology
- **output_dir** (*str*) – The directory in which to output the topology. (Default: None)
- **debug** (*bool*) – Enable debugging (Default: False)

gns3converter.main.**get_snapshots**(*topology*)

Return the paths of any snapshot topologies

Parameters **topology** (*str*) – topology file

Returns list of dicts containing snapshot topologies

Return type list

gns3converter.main.**main**()

Entry point for gns3-converter

gns3converter.main.**make_qemu_dirs**(*max_qemu_id*, *output_dir*, *topology_name*)

Create Qemu VM working directories if required

Parameters

- **max_qemu_id** (*int*) – Number of directories to create
- **output_dir** (*str*) – Output directory
- **topology_name** (*str*) – Topology name

gns3converter.main.**make_vbox_dirs**(*max_vbox_id*, *output_dir*, *topology_name*)

Create VirtualBox working directories if required

Parameters

- **max_vbox_id** (*int*) – Number of directories to create
- **output_dir** (*str*) – Output directory
- **topology_name** (*str*) – Topology name

gns3converter.main.**name**(*topology_file*, *topology_name=None*)

Calculate the name to save the converted topology as using either either a specified name or the directory name of the current project

Parameters

- **topology_file** (*str*) – Topology filename
- **topology_name** (*str or None*) – Optional topology name (Default: None)

Returns new topology name

Return type str

gns3converter.main.**save**(*output_dir*, *converter*, *json_topology*, *snapshot*, *quiet*)

Save the converted topology

Parameters

- **output_dir** (*str*) – Output Directory
- **converter** (Converter) – Converter instance
- **json_topology** (JSONTopology) – JSON topology layout
- **snapshot** (*bool*) – Is this a snapshot?
- **quiet** (*bool*) – No console printing

gns3converter.main.**setup_argparse**()
Setup the argparse argument parser

Returns instance of argparse

Return type ArgumentParser

gns3converter.main.**snapshot_name**(*topo_name*)
Get the snapshot name

Parameters **topo_name** (*str*) – topology file location. The name is taken from the directory containing the topology file using the following format: topology_NAME_snapshot_DATE_TIME

Returns snapshot name

Raises **ConvertError** when unable to determine the snapshot name

gns3converter.main.**topology_abspath**(*topology*)
Get the absolute path of the topology file

Parameters **topology** (*str*) – Topology file

Returns Absolute path of topology file

Return type str

gns3converter.main.**topology_dirname**(*topology*)
Get the directory containing the topology file

Parameters **topology** (*str*) – topology file

Returns directory which contains the topology file

Return type str

3.5 gns3converter.models

Convenience module for building a model matrix arranged by:

- Model
- Chassis (if applicable)

and containing:

- 'ports' = number of ports
- 'type' = type of ports

3.6 gns3converter.node

This module is used for building Nodes

class gns3converter.node.**Node**(*hypervisor*, *port_id*)
Bases: *gns3converter.interfaces.Interfaces*

This class defines a node used for building the Nodes configuration

Parameters

- **hypervisor** – Hypervisor
- **port_id** (*int*) – starting port ID for this node

add_device_items(*item*, *device*)
Add the various items from the device to the node

Parameters

- **item** (*str*) – item key
- **device** (*dict*) – dictionary containing items

add_info_from_hv()
Add the information we need from the old hypervisor section

add_mapping(*mapping*)

add_slot_ports(*slot*)
Add the ports to be added for a adapter card

Parameters **slot** (*str*) – Slot name

add_to_qemu()
Add additional parameters to a QemuVM Device that were present in its global conf section

add_to_virtualbox()
Add additional parameters that were in the VBoxDevice section or not present

add_vm_ethernet_ports()
Add ethernet ports to Virtualbox and Qemu nodes

add_wic(*old_wic*, *wic*)
Convert the old style WIC slot to a new style WIC slot and add the WIC to the node properties

Parameters

- **old_wic** (*str*) – Old WIC slot
- **wic** (*str*) – WIC name

add_wic_ports(*wic_slot*)
Add the ports for a specific WIC to the node['ports'] dictionary

Parameters **wic_slot** (*str*) – WIC Slot (wic0)

calc_cloud_connection()
Add the ports and nios for a cloud connection

Returns None on success or RuntimeError on error

calc_device_links()
Calculate a router or VirtualBox link

calc_ethsw_port(*port_num*, *port_def*)
Split and create the port entry for an Ethernet Switch

Parameters

- **port_num** (*str or int*) – port number

- **port_def** (*str*) – port definition

calc_frsw_port(*port_num*, *port_def*)
Split and create the port entry for a Frame Relay Switch

Parameters

- **port_num** (*str or int*) – port number
- **port_def** (*str*) – port definition

calc_link(*src_id*, *src_port*, *src_port_name*, *destination*)
Add a link item for processing later

Parameters

- **src_id** (*int*) – Source node ID
- **src_port** (*int*) – Source port ID
- **src_port_name** (*str*) – Source port name
- **destination** (*dict*) – Destination

calc_mb_ports()
Add the default ports to add to a router

get_nb_added_ports(*old_port_id*)
Get the number of ports add to the node

Parameters **old_port_id** (*int*) – starting port_id

Returns number of ports added

Return type int

process_mappings()
Process the mappings for a Frame Relay switch. Removes duplicates and adds the mappings to the node properties

set_description()
Set the node description

set_qemu_symbol()
Set the appropriate symbol for QEMU Devices

set_symbol(*symbol*)
Set a symbol for a device

Parameters **symbol** (*str*) – Symbol to use

set_type()
Set the node type

3.7 gns3converter.topology

This module is for processing a topology

class gns3converter.topology.**JSONTopology**
Bases: object

v1.0 JSON Topology

get_qemus()
Get the maximum ID of the Qemu VMs

Returns Maximum Qemu VM ID

Return type int

get_topology()
Get the converted topology ready for JSON encoding

Returns converted topology assembled into a single dict

Return type dict

get_vboxes()
Get the maximum ID of the VBoxes

Returns Maximum VBox ID

Return type int

images
Returns the images

Returns Topology images

Return type list

links
Returns the links

Returns Topology links

Return type list

name
Returns the topology name

Returns Topology name

Return type None or str

nodes
Returns the nodes

Returns topology nodes

Return type list

notes
Returns the notes

Returns Topology notes

Return type list

servers
Returns the servers

Returns Topology servers

Return type list

shapes
Returns the shapes

Returns Topology shapes

Return type dict

class gns3converter.topology.**LegacyTopology**(*sections*, *old_top*)
Bases: object

Legacy Topology (pre-1.0)

Parameters

- **sections** (*list*) – list of sections from gns3converter.converter.Converter.get_instances()
- **old_top** (*ConfigObj*) – Old topology as returned by *gns3converter.converter.Converter.read_topology()*

add_artwork_item(*instance*, *item*)
Add an artwork item e.g. Shapes, Notes and Pixmaps

Parameters

- **instance** – Hypervisor instance
- **item** – Item to add

add_conf_item(*instance*, *item*)
Add a hypervisor configuration item

Parameters

- **instance** – Hypervisor instance
- **item** – Item to add

add_physical_item(*instance*, *item*)
Add a physical item e.g router, cloud etc

Parameters

- **instance** – Hypervisor instance
- **item** – Item to add

add_qemu_path(*instance*)
Add the qemu path to the hypervisor conf data

Parameters **instance** – Hypervisor instance

artwork
Return the Artwork dict

Returns artwork dict

Return type dict

static device_typename(*item*)
Convert the old names to new-style names and types

Parameters **item** (*str*) – A device in the form of 'TYPE NAME'

Returns tuple containing device name and type details

hv_id
Return the Hypervisor ID

Returns Hypervisor ID

Return type int

nid
Return the node ID

Returns Node ID

Return type int

qemu_id
Return the Qemu VM ID :return: Qemu VM ID :rtype: int

vbox_id
Return the VBox ID :return: VBox ID :rtype: int

3.8 gns3converter.utils

gns3converter.utils.**fix_path**(*path*)
Fix windows path's. Linux path's will remain unaltered

Parameters **path** (*str*) – The path to be fixed

Returns The fixed path

Return type str

CHAPTER 4

Development

If you find a bug in gns3-converter please feel free to report it to the issue tracker listed below. If the problem occurs with a particular topology, please include the topology with the issue report.

- Public Repository: https://github.com/dlintott/gns3-converter
- Issue Tracker: https://github.com/dlintott/gns3-converter/issues
- License: GPL-3+

g

A

C

D

F

G

H

I

J

L

M

N

P

Q

R

S

T

V