



P4 Runtime Controller

Table of contents

Introduction

p4runtime_sh Usage

P4 info

P4 Table

Working with P4 Table Entries

Working with Direct Counters

Working with Indirect Counters

P4 Actions

Packet IO

DPL applications are deployed to the NVIDIA® BlueField® networking platform (DPU or SuperNIC) via the P4Runtime API.

Since DPL is derived from the P4-16 language, it is compatible with the P4Runtime specification, enabling standard runtime interaction with the compiled DPL pipeline.

Introduction

The [P4 Runtime shell](#) (`p4runtime_sh`) is an open-source CLI tool that provides an interface to the P4Runtime API. It is especially useful for:

- Loading simple DPL programs
- Testing match-action tables
- Debugging pipeline behavior

The shell can be invoked using the launch script provided in the DPL Development container.

Info

For detailed instructions, refer to [Loading DPL Applications](#).

p4runtime_sh Usage

The following are example commands for using the [p4runtime_sh P4 Controller](#) after loading a program.

P4 info


Operation	Command
Retrieves the content of <code>p4info.txt</code> of the currently loaded DPL program	<pre>p4inf</pre>

Operation	Command
	o

P4 Table

Operation	Command
Lists all P4 tables	tables
Displays information about a specific P4 table	tables["<P4_TABLE_NAME>"]

Working with P4 Table Entries

Operation	Command
Default Entry	
Reads P4 table's default entry without counter's value	<pre>te = table_entry[" <P4_TABLE_NAME>"] te.is_default = True te.read(lambda te: print(te))</pre>
Reads P4 table's default entry with counter's value	<pre>te = table_entry[" <P4_TABLE_NAME>"] te.is_default = True</pre>
<p> Note</p>	

Operation	Command
<p>Supported only if a direct counter is enabled on the P4 table.</p>	<pre>te.counter_data.byte_count = 0 te.read(lambda te: print(te))</pre>
<p>Modifies P4 table's default entry action</p> <div data-bbox="159 688 992 951" style="background-color: #ffffcc; padding: 10px;"> <p>Note A default entry cannot be removed or inserted. It can only be modified to perform a different P4 action.</p> </div>	<pre>te = table_entry[" <P4_TABLE_NAME>"] (action=" <P4_ACTION_NAME>") te.is_default = True # Set value for all parameters required by desired action. te.action["<PARAMETER NAME>"] = "<PARAMETER VALUE>" te.modify()</pre>
<p>Regular Entries</p>	
<p>Note: In the following examples, some commands require specifying the match key of the desired regular P4 table key. The syntax for specifying a match key varies according to the defined match method for each key (per the DPL source code where the keys are defined on the P4 table).</p>	
Match Method	Syntax for Specifying Match Key Value
exact	<pre>te.match["<MATCH_KEY_NAME>"] = "<MATCH_VALUE>"</pre>
ternary	<pre>te.match["<MATCH_KEY_NAME>"] = "<MATCH_VALUE>&&&<MASK_VALUE>" te.priority = <PRIORITY VALUE></pre>

Operation	Command
Match Method	Syntax for Specifying Match Key Value
	Note that mask is provided in the match line, separated by &&& . If mask is not specified , a full match mask will be used.
lpm	<pre data-bbox="358 464 1450 562">te.match["<MATCH_KEY_NAME>"] = "<MATCH_VALUE>/<PREFIX_LENGTH_VALUE>"</pre> <p data-bbox="318 575 1414 653">Note that LPM prefix_len is provided in the match line, separated by /. If LPM prefix_len is not specified , a prefix_len with full field bitwidth is used.</p>
For simplicity, the following examples are written using <code>exact</code> match syntax.	
Reading Entries	
Reads a specific regular P4 table entry	<pre data-bbox="1073 856 1446 1787">te = table_entry[" <P4_TABLE_NAME>"] try: pass # Comment out the next line to disable reading counters te.counter_data.byte_count = 0 except Exception as e: # Table does not have a Direct Counter pass # Set value for all keys required by the P4 table. te.match[" <MATCH_KEY_NAME>"] = " <MATCH_VALUE>"</pre>

Operation	Command
	<pre>te.read(lambda te: print(te))</pre>
<p>Reads all regular entries from a P4 table</p>	<pre>num = 1 def hndlr(te): global num print(f">> Entry number {num}:") print(te) print("----- -----") num += 1 te = table_entry[" <P4_TABLE_NAME>"] try: pass # Comment out the next line to disable reading counters te.counter_data.byte e_count = 0 except Exception as e: # Table does not have a Direct Counter pass # Read regular entries te.is_default = False</pre>

Operation	Command
	<pre>te.read(lambda te: hndlr(te))</pre>
<p>Reads all regular entries from all P4 tables in the P4 program</p>	<pre>for tbl in tables: num = 1 def hndlr(te): global num print(f">> Entry number {num}:") print(te) print("----- -----") num += 1 print(f"===== {tbl.name} =====") te = table_entry[tbl.name] try: pass # Comment out the next line to disable reading counters te.counter_data.byte e_count = 0 except Exception as e: # Table does not have a Direct Counter pass</pre>

Operation	Command
	<pre data-bbox="1073 275 1414 506"> # Read regular entries te.is_default = False te.read(lambda te: hndlr(te)) </pre>
Adding Entries	
<p data-bbox="159 1031 586 1066">Adds a regular P4 table entry</p>	<pre data-bbox="1073 709 1438 1388"> te = table_entry[" <P4_TABLE_NAME>"] (action=" <P4_ACTION_NAME>") # Set value for all keys required by the P4 table. te.match[" <MATCH_KEY_NAME>"] = " <MATCH_VALUE>" # Set value for all parameters required by desired action. te.action["<PARAMETER NAME>"] = "<PARAMETER VALUE>" te.insert() </pre>
Deleting Entries	
<p data-bbox="159 1535 748 1570">Deletes a specific regular P4 table entry</p>	<pre data-bbox="1073 1598 1422 1902"> te = table_entry[" <P4_TABLE_NAME>"] # Set value for all keys required by the P4 table. te.match[" <MATCH_KEY_NAME>"] = " <MATCH_VALUE>" </pre>

Operation	Command
	<pre>te.delete()</pre>
Deletes all regular entries from a P4 table	<pre>te = table_entry[" <P4_TABLE_NAME>"] te.read(lambda te: te.delete())</pre>

Working with Direct Counters

This allows working with Direct Counter directly without getting the whole Table Entry info.

Operation	Command
Lists all defined Direct Counters	<pre>direct_counters</pre>
Default Entry	
Reads counter data of a default entry in a Direct Counter	<pre>DirectCounterEntry(" <P4_DIRECT_COUNTER_NAME>") ce.table_entry.is_default = True ce.read((lambda ce: print(ce)))</pre>
Clears counter data of a default entry in a Direct Counter	<pre>DirectCounterEntry(" <P4_DIRECT_COUNTER_NAME>")</pre>

Operation	Command
	<pre>ce.table_entry.is_default = True ce.packet_count = 0 ce.modify()</pre>
Regular Entries	
<p>Reads counter data of a specific P4 table entry in a Direct Counter</p>	<pre>ce = DirectCounterEntry(" <P4_DIRECT_COUNTER_NAME>") # Set value for all keys required by the P4 table. ce.table_entry.match[" <MATCH_KEY_NAME>"] = " <MATCH_VALUE>" ce.read((lambda ce: print(ce)))</pre>
<p>Reads all counter data of specific Direct Counter</p> <div data-bbox="159 1234 841 1497" style="background-color: #ffffcc; padding: 10px;"> <p>i Info This will also read the default entry counter data.</p> </div>	<pre>ce = DirectCounterEntry(" <P4_DIRECT_COUNTER_NAME>") ce.read((lambda ce: print(ce)))</pre>
<p>Clears counter data of a a specific P4 table entry in a Direct Counter</p>	<pre>ce = DirectCounterEntry(" <P4_DIRECT_COUNTER_NAME>") # Set value for all keys required by the P4 table. ce.table_entry.match[" <MATCH_KEY_NAME>"] = "</pre>

Operation	Command
	<pre><MATCH_VALUE>" ce.packet_count = 0 ce.modify()</pre>
<p>Clears all counter data of specific Direct Counter</p> <p>Note This also clears the default entry counter data.</p>	<pre>ce = DirectCounterEntry(" <P4_DIRECT_COUNTER_NAME>") ce.packet_count = 0 ce.modify()</pre>
<p>Clears all direct counters from all table</p>	<pre>for dc in direct_counters: ce = DirectCounterEntry(dc.name) ce.packet_count = 0 ce.modify()</pre>

Working with Indirect Counters

Operation	Command
<p>Lists all defined Indirect Counters</p>	<pre>counters</pre>
<p>Shows info about a specific Indirect Counter</p>	<pre>counter_entry[" <P4_INDIRECT_COUNTER_NAME>"]</pre>

Operation	Command
Reads a specific value from a specific indirect counter	<pre>ce = counter_entry[" <P4_INDIRECT_COUNTER_NAME>"] ce.index = <COUNTER_CELL_INDEX> ce.read((lambda ce: print(ce)))</pre>
Reads all values from a specific indirect counter	<pre>ce = counter_entry[" <P4_INDIRECT_COUNTER_NAME>"] ce.read((lambda ce: print(ce)))</pre>
Clears a specific value from a specific indirect counter	<pre>ce = counter_entry[" <P4_INDIRECT_COUNTER_NAME>"] ce.index = <COUNTER_CELL_INDEX> ce.byte_count = 0 ce.packet_count = 0 ce.modify()</pre>
Clears all values from a specific indirect counter	<pre>ce = counter_entry[" <P4_INDIRECT_COUNTER_NAME>"] ce.byte_count = 0 ce.packet_count = 0 ce.modify()</pre>

P4 Actions

Operation	Command
Lists all defined P4 actions	<pre>actions</pre>
Shows info about a specific P4 actions	<div style="background-color: #ffffcc; padding: 5px;"> <p>i Info This includes its parameters' names and sizes.</p> </div> <pre>actions[" <P4_ACTION_NAME>"]</pre>

Packet IO

To use packet IO, it must be enabled in the P4 program source code.

Operation	Command
Packet In – For receiving packets sent from the DPL Runtime daemon to the P4 Controller (according to defined rules in the DPL program)	
Captures packets for 10 second, then displays them	<pre>packet_in.sniff(lambda m: print(m), timeout=10)</pre>
Capturs packets for 10 second, then displays them parsed as well as their metadata info	<p>This example command uses the <code>impacket</code> package for parsing the packets, so make sure that it is installed on your system prior to running the <code>p4runtime_sh</code> P4 Controller:</p> <pre>pip install impacket</pre> <p>Then, from the <code>p4runtime_sh</code> P4 Controller, run:</p>

Operation	Command
	<pre> from impacket.ImpactDecoder import * for msg in packet_in.sniff(timeout=10): print("-----") print(msg) print("+++++") print("Raw packet (hex):\n") print(msg.packet.payload.hex()) print("+++++") print("Parsed packet:\n") print(EthDecoder().decode(msg.packet.payload)) print("+++++") print("Metadata:\n") for md in msg.packet.metadata: val = int.from_bytes(md.value, "big") print("metadata ID (" + md.metadata_id + "):", hex(val)) </pre>
<p>Packet Out – For sending a packet from the <code>p4runtime_sh</code> P4 Controller to the DPL Runtime daemon (which will process it according to defined rules in the DPL program)</p>	
<p>Syntax of sending a packet</p>	<pre> my_pkt = b'<PACKET_BYTE_STRING_MESSAGE_GOES_HERE>' </pre>

Operation	Command
	<pre>packet_out(payload=my_pkt, <METADATA_NAME>="METADATA_VALUE_GOES_HERE") . send() Repeat the <METADATA_NAME>= "METADATA_VALUE_GOES_HERE " parameter for each defined metadata in the P4 program.</pre>
<p>Example of building and sending a packet</p>	<p>Using <code>scapy</code> tool, build the desired packet:</p> <pre>>>> p = Ether(src='00:11:11:11:11:11', dst='00:22:22:22:22:22') / IP(src="1.1.1.1", dst="2.2.2.2") >>> print(p.build()) b'\x00\x00\x00\x00\x11\x11\x11\x11\x11\x11\x08\x00E\x00\x00\x14\x00\x01\x00\x00@\x00t\xe4\x01\x01\x01\x01\x02\x02\x02\x02' >>></pre> <p>Then send it using the <code>p4runtime_sh</code> P4 Controller:</p> <pre>my_pkt = b'\x00\x00\x00\x00\x11\x11\x11\x11\x11\x11\x08\x00E\x00\x00\x14\x00\x01\x00\x00@\x00t\xe4\x01\x01\x01\x01\x02\x02\x02\x02' packet_out(payload=my_pkt, controller_metadata="0x123") . send()</pre>

Notice
This document is provided for information purposes only and shall not be regarded as a warranty of a certain functionality, condition, or quality of a product. NVIDIA Corporation ("NVIDIA") makes no representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assumes no responsibility for any errors contained herein. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This document is not a commitment to develop, release, or deliver any Material (defined below), code, or functionality.
NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice.
Customer should obtain the latest relevant information before placing orders and should verify that such information is current and complete.
NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer ("Terms of Sale"). NVIDIA hereby expressly objects to applying any customer general terms and conditions with regards to the purchase of the NVIDIA product referenced in this document. No contractual obligations

are formed either directly or indirectly by this document. NVIDIA products are not designed, authorized, or warranted to be suitable for use in medical, military, aircraft, space, or life support equipment, nor in applications where failure or malfunction of the NVIDIA product can reasonably be expected to result in personal injury, death, or property or environmental damage. NVIDIA accepts no liability for inclusion and/or use of NVIDIA products in such equipment or applications and therefore such inclusion and/or use is at customer's own risk. NVIDIA makes no representation or warranty that products based on this document will be suitable for any specified use. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer's sole responsibility to evaluate and determine the applicability of any information contained in this document, ensure the product is suitable and fit for the application planned by customer, and perform the necessary testing for the application in order to avoid a default of the application or the product. Weaknesses in customer's product designs may affect the quality and reliability of the NVIDIA product and may result in additional or different conditions and/or requirements beyond those contained in this document. NVIDIA accepts no liability related to any default, damage, costs, or problem which may be based on or attributable to: (i) the use of the NVIDIA product in any manner that is contrary to this document or (ii) customer product designs. No license, either expressed or implied, is granted under any NVIDIA patent right, copyright, or other NVIDIA intellectual property right under this document. Information published by NVIDIA regarding third-party products or services does not constitute a license from NVIDIA to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property rights of the third party, or a license from NVIDIA under the patents or other intellectual property rights of NVIDIA. Reproduction of information in this document is permissible only if approved in advance by NVIDIA in writing, reproduced without alteration and in full compliance with all applicable export laws and regulations, and accompanied by all associated conditions, limitations, and notices. THIS DOCUMENT AND ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL NVIDIA BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA's aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms of Sale for the product. **Trademarks** NVIDIA and the NVIDIA logo are trademarks and/or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

© Copyright 2025, NVIDIA. PDF Generated on 04/24/2025