Network Flow Processor SDK

NFP SDK 6.0.4.1

Release Notes

- Proprietary and Confidential -

b3794.dr7171

Product code
080-00004-012

Network Flow Processor SDK: Release Notes

COPYRIGHT

WARRANTY

Netronome warrants that any media on which this documentation is provided will be free from defects in materials and workmanship under normal use for a period of ninety (90) days from the date of shipment. If a defect in any such media should occur during this 90-day period, the media may be returned to Netronome for a replacement.

NETRONOME DOES NOT WARRANT THAT THE DOCUMENTATION SHALL BE ERROR-FREE. THIS LIMITED WARRANTY SHALL NOT APPLY IF THE DOCUMENTATION OR MEDIA HAS BEEN (I) ALTERED OR MODIFIED; (II) SUBJECTED TO NEGLIGENCE, COMPUTER OR ELECTRICAL MALFUNCTION; OR (III) USED, ADJUSTED, OR INSTALLED OTHER THAN IN ACCORDANCE WITH INSTRUCTIONS FURNISHED BY NETRONOME OR IN AN ENVIRONMENT OTHER THAN THAT INTENDED OR RECOMMENDED BY NETRONOME.

EXCEPT FOR WARRANTIES SPECIFICALLY STATED IN THIS SECTION, NETRONOME HEREBY DISCLAIMS ALL EXPRESS OR IMPLIED WARRANTIES OF ANY KIND, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE.

Some jurisdictions do not allow the exclusion of implied warranties, so the above exclusion may not apply to some users of this documentation. This limited warranty gives users of this documentation specific legal rights, and users of this documentation may also have other rights which vary from jurisdiction to jurisdiction.

LIABILITY

Regardless of the form of any claim or action, Netronome's total liability to any user of this documentation for all occurrences combined, for claims, costs, damages or liability based on any cause whatsoever and arising from or in connection with this documentation shall not exceed the purchase price (without interest) paid by such user.

IN NO EVENT SHALL NETRONOME OR ANYONE ELSE WHO HAS BEEN INVOLVED IN THE CREATION, PRODUCTION, OR DELIVERY OF THE DOCUMENTATION BE LIABLE FOR ANY LOSS OF DATA, LOSS OF PROFITS OR LOSS OF USE OF THE DOCUMENTATION OR FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, EXEMPLARY, PUNITIVE, MULTIPLE OR OTHER DAMAGES, ARISING FROM OR IN CONNECTION WITH THE DOCUMENTATION EVEN IF NETRONOME HAS BEEN MADE AWARE OF THE POSSIBILITY OF SUCH DAMAGES. IN NO EVENT SHALL NETRONOME OR ANYONE ELSE WHO HAS BEEN INVOLVED IN THE CREATION, PRODUCTION, OR DELIVERY OF THE DOCUMENTATION BE LIABLE TO ANYONE FOR ANY CLAIMS, COSTS, DAMAGES OR LIABILITIES CAUSED BY IMPROPER USE OF THE DOCUMENTATION OR USE WHERE ANY PARTY HAS SUBSTITUTED PROCEDURES NOT SPECIFIED BY NETRONOME.

Revision History

Date	Revision	Description
30 July 2018	012	Release of NFP SDK 6.0.4.1
30 April 2018	011	Release of NFP SDK 6.0.4
7 July 2017	010	Release of NFP SDK 6.0.3
31 March 2017	009	Release of NFP SDK 6.0.2
3 October 2016	008	Release of NFP SDK 6.0.1
27 June 2016	007	Release of NFP SDK 6.0.0
23 May 2016	006	Release of NFP SDK 6.0 Beta 1
4 February 2016	005	Release of NFP SDK 6.0 Managed C Beta 1
31 December 2015	004	Release of NFP SDK 6.0-preview
30 September 2015	003	Release of NFP SDK 5.2.0
30 September 2014	002	Release of NFP SDK 5.1.0
31 March 2014	001	Release of NFP SDK 5.0.0 Beta

Table of Contents

List of Tables

1. Introduction

1.1 Purpose of Release

This is the 6.0.4.1 release of the Netronome® Network Flow Processor Software Development Kit (SDK).

This release supports P4, Managed C (experimental), Micro-C and Microcode development on the NFP-4xxx and NFP-6xxx family of devices and provides full support for developing and debugging software utilizing the 120 Flow Processing Cores offered by the architecture.

This release improves on prior releases by including several fixes and enhancements as described in section 4 of this document.

1.2 Overview

The SDK consists of the following components:

- Integrated Development Environment (Programmer Studio)
- P4 Tools (nfp4c, nfirc, nfp4build) ✓
- Microcode Assembler (nfas) ✓
- Micro-C Compiler (nfcc) ✓
- Microcode and Micro-C library ✓
- Linker (nfld) and Loader (nfp-nffw) ✓
- NFP-4xxx/NFP-6xxx Simulator (nfsim) ✓
- Hardware Debug Server (nfp-sdk-hwdbgsrv) ✓
- Run Time Environment/RTE (pif_rte) ✓

Note
✓ indicates, in addition to Windows® a Linux® version is also available.

Note
To use Programmer Studio, a license key from Netronome Support should be obtained by emailing support@netronome.com.

Note
The RTE is only needed for developing and running P4 and Managed C applications.

The following table provides an overview of the languages with development models available in this release:

Table 1.1. Comparison of language with development model

	P4	Managed C (experimental)	Micro-C	Microcode
Development Effort	Least	More than P4	Easier than microcode	Most
Expertise needed	P4 language knowledge	Micro-C and some level of NFP Architecture	Micro-C and detailed knowledge of NFP Architecture	Microcode and detailed knowledge of NFP Architecture
Dataplane	Included	Included	DIY	DIY
Protocols	Defined as P4 language constructs	Typically defined in C structs	Typically defined in C structs	Typically defined in bitfield macros
Packet Delivery	Extracted headers with packet metadata	Raw packet with data starting at Ethernet header and metadata	NA	NA
Packet Processing Speed	Good	Good	Good	Depending on developer proficiency can be the most
Number of Flow Processing Cores/(Threads) available for user code (depends on NFP variant)	up to 70/(560)	up to 70/(560)	up to 120/(960)	up to 120/(960)
Matching	LPM, ternary, exact, range, index, valid	DIY	DIY	DIY
Flow cache for actions	Included, based on flowkey derived from extracted headers	DIY	DIY	DIY
Packet ordering	Maintained	Maintained	DIY	DIY
Supported Ports	Network and SR-IOV VFs on host	Network and SR-IOV VFs on host	DIY	DIY
Host delivery	SR-IOV VF delivery, selected via metadata	SR-IOV VF delivery, selected via metadata	DIY or NFD	DIY or NFD
Port initialization, VF bring up	Supported using RTE	Supported using RTE	DIY, can use BSP tools like mac_init etc.	DIY, can use BSP tools like mac_init etc.
C Sandbox/plugin	Using P4 primitive_action	Not needed, can do C function call	Not needed, can do C function call	NA, can run C code on other MEs
Debugging on Simulator	Supported but SR-IOV not supported	Supported but SR-IOV not supported	Supported but SR-IOV not supported	Supported but SR-IOV not supported

	P4	Managed C (experimental)	Micro-C	Microcode
Debugging on Hardware	Supported using debug server and RTE	Supported using debug server and RTE	Supported using debug server	Supported using debug server

1.3 Installation

- The Windows® hosted edition of the SDK is installed by downloading the installer executable and running it on the machine where the SDK should be installed. The installer executable has a name like `nfp-sdk-<release number>-setup.exe`.

  It installs the P4 tools, Micro-C compiler, Microcode Assembler, Microcode and Micro-C Library, Linker, RTE for use with the NFP Simulator, Loader and the NFP Simulator.

  The installer will prompt and provide guidance through the installation procedure.

  

  Note
  The simulator requires a 64-bit CPU (and operating system) and if this is not detected during installation the simulator will not be installed.

- The Linux hosted NFP SDK tool chain is available as an Ubuntu or CentOS/RHEL package or as a tarball. The Ubuntu/CentOS/RHEL package installs the tool chain into the `/opt/netronome/bin` directory for use on a host where the BSP is installed (when developing on an Agilio SmartNIC).

  On Ubuntu the tool chain is installed with the command:

  ```
  dpkg -i nfp-sdk-<release number>.x86_64.deb
  ```

  and on CentOS/RHEL with:

  ```
  rpm -i nfp-sdk-<release number>.x86_64.rpm
  ```

  The tarball `nfp-sdk-<release number>.x86_64.tgz` is for development with the NFP simulator. (In addition to the tool chain the tarball contains libraries for communicating with the NFP simulator).

  The tool chain and simulator are installed into the subdirectory `nfp-sdk-<release number>` (under the current working directory) with the following commands:

  ```
  tar xvzf nfp-sdk-<release number>.x86_64.tgz

  tar xvzf nfp-sdk-sim-<release number>.x86_64.tgz
  ```

- The Run Time Environment (RTE) needs to be installed and started on the x86_64 server hosting the Agilio SmartNIC. Currently packages are available for Ubuntu and CentOS/RHEL and is installed by extracting the RTE using the following command:

  ```
  tar xvzf nfp-sdk-p4-rte-<release number>.[ubuntu|centos].x86_64.tgz
  ```

In addition to the RTE the package contains the BSP and the source code for the Hardware Debug Server (for use with Programmer Studio). The debug server is built during installation.

The recommened installation procedure on a system with a Netronome SmartNIC is to install the RTE with the included BSP packages with `./sdk6_rte_install.sh install`

This is the recommended installation procedure for using the **SDK6 RTE with SDK6 simulator** on Linux:

- To install only the RTE without the NFP BSP with `./sdk6_rte_install.sh install_rte_only`
- Download the latest release tarball of the SDK simulator matching the RTE version and extract the tarball to a desired location, a directory nfp-sdk-6.x.x will be created into which all the files will be extracted. This directory will be SDK6 simulator installation directory, for example if simulator version 6.0.3 is extracted to /opt the installation directory will be /opt/nfp-sdk-6.0.3.
- From the installation directory go to examples/nfsim and run make.
- Create symlinks to nfp and nfp_common naming compatible with NFP BSP libraries in the lib directory in the installation directory

```
cd /opt/nfp-sdk-6.0.3/lib
ln -s libnfp.so libnfp.so.3
ln -s libnfp_common.so libnfp_common.so.0
```

  Before running the RTE the lib directory in installation directory should be added to LD_LIBRARY_PATH and the environmental variable NETRODIR should be set to the installation directory, example:

```
export LD_LIBRARY_PATH=/opt/nfp-sdk-6.0.3/lib;LD_LIBRARY_PATH
export NETRODIR=/opt/nfp-sdk-6.0.3
```

  Start the simulator by running nfsim in the bin subdirectory before starting the RTE in SIM MODE.

The `README` file in the top level directory contains more detailed instructions for the RTE installation and running.

Note

When installing a new SmartNIC after the initial RTE installation, the installation script should be rerun with `./sdk6_rte_install install_force_bsp` to ensure the NFP BSP related firmware on the SmartNIC is up to date. This can also be done using NFP BSP tools, please refer to NFP BSP documentation for more information.

1.4 Resources

- The NFP community portal with tutorials and examples is available at *http://www.open-nfp.org*
- WHITE PAPER: Programming Netronome Agilio® Intelligent Server Adapters
- Netronome Flow Processor Linux kernel drivers: *https://github.com/Netronome/nfp-drv-kmods*

- Using DPDK with the Netronome Poll Mode Driver: *http://www.dpdk.org/doc/guides/nics/nfp.html*

1.5 Terminology

Acronym	Description
Agilio®	Netronome's family of SmartNIC's and Software
ARI	Alternate Routing ID Interpretation, a mechanism for a PCIe component to support more than 8 functions
BSP	NFP Board Support Package
CDP	Customer Development Platform
DIY	Do It Yourself
DPDK	Intel© Dataplane Development Kit
FPC	Flow Processing Core (also known as ME)
GUI	Graphical User Interface
SmartNIC	Agilio range of Inteligent Server Adapter
LPM	Longest Prefix Match
LTS	Long Term Support
ME	Microengine (also known as FPC)
NFAS	Network Flow Assembler
NFCC	Network Flow C Compiler
NFD	Netronome Flow Driver
NFFW	Network Flow Firmware (ELF file which can be loaded into the NFP)
NFLD	Network Flow Linker
NFP	Netronome Network Flow Processor
NIC	Network Interface Card
OS	Operating System
P4	P4 programming language for programming the data plane of network devices
PCIe	Peripheral Component Interconnect Express
PMD	Poll Mode Driver
PIF	Protocol Independent Forwarding
RTE	Run Time Environment
SR-IOV	Single Root I/O Virtualization
VF	SR-IOV Virtual Function
VM	Virtual Machine
WINE	Wine Is Not an Emulator - refer to https://www.winehq.org

2. Prerequisites and Supported Hardware

2.1 Windows Hosted Development Environment

The Windows host development system requires the following to support this release:

- **Minimum recommended system CPU:** 2.2 GHz multi core x86_64 CPU. Simulator performance will benefit from having a faster CPU with many cores.

> **Note**
> The simulator requires a 64-bit CPU (and operating system) to run. If a 64-bit environment is not present during installation the simulator will not be installed.

- **Minimum recommended memory:** At least 10GB of RAM with 2GB of virtual memory. The NFP-6xxx supports 24GB external memory and memory usage will depend on the memory usage of the application being simulated.
- **Hard disk space required:** 1GB should be free prior to installation.
- **Graphics card:** Minimum requirement is dictated by the host OS. A high resolution screen is recommended when using Programmer Studio for interactive debugging.
- **Supported host operating systems:** Windows 7, 8 or 10.
- **Hardware debugging:** For hardware debugging, version 139 or greater of the NFP-4xxx/NFP-6xxx debug server software is required.
- **Dependencies on other pre-installed software and their versions:** The base OS is sufficient to run the software. Adobe Acrobat, Adobe Reader or other PDF viewing software will be needed to read the documentation provided with the installation.
- **Dependencies on older versions:** Prior versions do not need to be installed to use this software. This release can be installed concurrently with 4.x releases.
- **Compatibility with virus scanners and firewalls:** Most virus scanners and firewalls should be compatible with this software. Firewalls need to be disabled when using the hardware debugging feature.
- **Networking:** An Internet connection is not required to use this software. For hardware debugging a network connection is needed between the host running Programmer Studio and the target containing the NFP.

2.2 Linux Hosted Development Environment

When development from a Linux host is prefered the system requirements for running the tools and simulator are:

- **Minimum recommended system CPU:** 2.2 GHz 64-bit
- **Minimum recommended memory:** At least 16GB of RAM with 2GB of virtual memory.

- **Supported host operating systems:** 64-bit Red Hat Enterprise Linux/CentOS 7 or Ubuntu 14.04/16.04 system. The tools should however function on other Linux distributions (e.g. Fedora) as well.

2.2.1 WINE support for Programmer Studio

Programmer Studio is not available as a native Linux application but can be used with WINE (Refer to https://www.winehq.org)

For Linux WINE 1.8 (or higher) and on OSX under WINE 1.9 (or higher) is recommended.

- On Ubuntu WINE can be installed by running:

  ```
  sudo apt-get install wine1.8
  ```

- On OSX installer package for "WINE Development" version 1.9 or higher can be downloaded from WINE HQ. WINE can be installed by running following package:

  ```
  winehq-devel-1.9.x.pkg
  ```

> **Note**
> When installing WINE on OSX, make sure to select **64 bit support (experimental)** check box on **Installation Type** page of WINE installer.

> **Note**
> The minimum supported version of Windows for Programmer Studio is Windows 7. This can be set for WINE by running `winecfg` and selecting it on the **Applications** page.

To install Programmer Studio, download the installer and run:

```
wine nfp-sdk-6.x.x.x-setup.exe
```

By default it will be installed to `~/.wine/drive_c/NFP_SDK_6.x.x` and can started with:

```
wine ~/.wine/drive_c/NFP_SDK_6.x.x/bin/Programmer_Studio.exe
```

2.3 Agilio Host Requirements

The server hosting the Agilio SmartNIC has the following requirements:

- **Minimum recommended system CPU:** A CPU with IO virtualization support is needed when using VFs.
- **Minimum recommended memory:** Memory requirements will typically be driven by the OS and whether VMs and/or DPDK is being used. 256MB for the RTE and debug server alone would be sufficient.
- **Supported host operating systems:** The RTE and debug server require a 64-bit Red Hat Enterprise Linux 7, CentOS 6/7 or Ubuntu 14.04/16.04/18.04 system. Both are expected to function on other distributions as well.

RTE installation script only supports Ubuntu 14.04, 16.04 and 18.04; CentOS 6.x and 7.x and RHEL 7.x. If SDK RTE needs to be installed on another version or distribution it will need to be done manually.

- **Compatibility with firewalls:** Firewalls need to be disabled or TCP port 20206 and 20406 needs to be set to accept inbound connections for the RTE and debug server respectively. The port numbers can be changed if needed.
- **Host system requirements:** Other than for the CPU, the system board also needs to support SR-IOV and ARI when VFs are used. Refer to the host system BIOS and manufacturer documentation to determine if this feature is supported by the server and what the required configuration options are to enable SR-IOV.

Note

The `/opt/netronome/bin/nfp-support` command can be used to determine if the server has the necessary support.

Note

There are Linux distributions and particular BIOS combinations that may require additional kernel command line options to enable SR-IOV, e.g. the pci=realloc kernel parameter. Refer to the kernel and/or distribution documentation for such requirements.

- **Patch level requirements:** Older Linux kernels exhibit undesired behavior in PCIe configuration code which is described in ERR47. Netronome submitted a fix to the kernel maintainers for this issue which has been accepted into kernel version 4.5. When using an older kernel version a patch needs to be applied to the kernel source code and the patched kernel needs to be installed. For convenience, packages containing patched versions of the default kernels for various Ubuntu 14.04 LTS and RHEL/CentOS 7 variants are available on Netronome's support site (http://support.netronome.com).

Note

The `err47_check.sh` script can be used to check if ERR47 is an issue. If the return code is not 0 the patch needs to be applied.

The following command can be run from a bash shell to indicate the status:

```
if /opt/nfp_pif/scripts/err47_check.sh; then echo "Kernel is good"; else
echo "Kernel patch needed"; fi
```

Sample .deb and .rpm packages, on Netronome support site, are as follows:

- Ubuntu 14.04.3 (kernel 3.19.0-33)
- Ubuntu 14.04.2 (kernel 3.16.0-53)
- RHEL/CentOS 7.1 (kernel 3.10.0-229.14.1.el7.x86_64)

Please refer to the Netronome support site as these sample packages will be updated from time to time for more recent kernel versions.

To install, transfer the package files to a subdirectory on your system, change to that subdirectory, and enter:

- `dpkg -i *.deb` (on Debian systems), or:
- `rpm -Uvh *.rpm` (on RHEL/CentOS systems).

Contact Netronome support if additional assistance is required.

Software for kernel versions can be found at https://www.kernel.org/. In the event software cannot be retrieved, contact Netronome via support site (http://support.netronome.com) for additional assistance in obtaining.

This kernel patch code is free software; you can redistribute it and/or modify it under the terms of version 2 of the GNU General Public License as published by the Free Software Foundation.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details, a copy of this license has been provided in the accompanying README file.

3. General

3.1 Programmer Studio Integrated Development Environment

- Programmer Studio supports P4, Managed C (experimental), Micro-C and Microcode application development.

  For P4 and Managed C applications a dataplane is constructed for basic packet transmission. The dataplane supports physical ports (on the Agilio-LX and CX SmartNICs) and SR-IOV VFs on the host.

> **Note**
> The port numbering for the second port on the Agilio® 2x10GbE is 1 as in releases prior to 6.0.2

> **Note**
> To create a P4 or a managed C mode application the appropriate checkbox must be selected when creating a new project.

> **Note**
> P4 Language specification 1.1 is selectable but in this release its use is still experimental.

> **Note**
> A host with an Agilio-LX or CX SmartNIC must support ARI (Alternative Routing ID) for SR-IOV VFs to be functional.

- In this release its possible to select an Agilio SmartNIC as the project target. Product numbers start with an `AMDA` prefix. Details about the products listed in the *Chip Selection* dialog are available in Appendix A. Also, selecting a product number will enable tooltip for *Chip Selection* dialog which gives a brief information for the selected SmartNIC. Selecting a product number will configure the project for the chip, memory and clock frequency of the SmartNIC.
- The number of worker MEs running a P4 or Managed C application can be selected from the *P4/Managed C Build Settings* dialog. Depending on the NFP variant and whether building for the simulator or hardware the range is between 1 and 70. (When set to 1, the application will run on the master ME, i32:0).
- The expected format of input fields for the *P4 User Config* dialogs are shown in the following table:

Table 3.1. P4 Input Field Format

Data Field	Format	Range	Example
Decimal Integer			123456789
Hexadecimal Integer			0x123456789

Data Field	Format	Range	Example
Drop Port			drop
Physical Network Port	p<N>	0-23	p0
SR-IOV VF	v0.<N>	0-31	v0.0
Multicast Group Name	mg<N>	0-15	mg0
IP			10.0.0.100
IP with Mask			10.0.0.100/32
IPv6			1234:::5678
IPv6 with Mask			1234:::5678/64
Range Match	<N>-><M>	0-65536	45->127
Valid Value			valid, invalid

Note

The *P4 User Config* dialogs depend on the constructs/parameters in the P4 source code, it is thus necessary to Build before the dialogs will reflect the code.

When rules are added or updated while in a debug session they will be sent to the RTE for installation in the datapath when saved.

- If an opened source file in Programmer Studio's editor contains a *#include* directive and you want to open the file, **Open Document** menu item has been added to the shortcut menu when user right click on the line that contains *#include* directive.
- Programmer Studio does not support accessing files directly via network shares, e.g. `\\servername\directory\filename`. Network files can be accessed if drive letters have been mapped to the shares by using a path which contains the drive letter, e.g. `z:\directory\filename`. Similarly Programmer Studio should not be installed into a directory on the network unless a drive letter has been mapped and the drive letter is used to access the installation directory.
- When NFP SDK 4.x or 5.x projects are opened and saved using Programmer Studio, they will be upgraded to NFP SDK 6 projects. Earlier versions of Programmer Studio will not be able to open the upgraded project files. To work around this behavior, do not save the project, but use the **Save Project As** option on the **File** menu to save it under another name.
- When adding additional toolbars to Programmer Studio, the default is to let the toolbar extend beyond the edge of the screen. To restore access to hidden icons, configure the toolbar to be displayed in free-floating mode.
- When in simulation mode, the packet streaming dialog for Agilio-CX, Hydrogen and Beryllium (1x40GE, 2x40GE or 4x10GE) shows more Ethernet ports than exist on the hardware. The additional ports should not be used.

3.2 P4 Programming Language

- This release supports language specification 1.0.2 with the limitations listed in Section 3.2.2.

Note
P4 Language specification 1.1 is selectable but its use is still experimental.

- P4 Applications also support Micro-C code running in a C Sandbox. Sandbox code can be invoked from P4 actions and will execute in the context of the action code.

3.2.1 C Sandbox

- For running a C Sandbox in the P4 dataplane please refer to the Appendix in the *Development Tools Users Guide* for more information.
- When changing a sandbox function name PS doesn't always detect the change and doesn't update the dataplane. When build errors are encountered doing a rebuild (ALT+F7) usually resolves it.

3.2.2 General P4 limitations

- It is important to note that compound actions are executed in series, as opposed to the parallel as the 1.0 P4 specification dictates.
- In this release the limit on the number of clones is one (1).
- Arbitrary LPM matching is not supported.
- Value sets currently don't use the mask field.
- Variable length fields can only be the last field in the header definition.
- It is possible that large P4 designs will hit either code store or various memory limitations within the NFP. Programmer Studio supports an option to build the application with reduced thread usage which will reduce resource requirements (at the possible expence of lower throughput). Refer to Appendix F in the *Developer Tools Users Guide* for more detail.

3.2.3 P4 stateful memory limitations

- Due to the cached lookups used for the NFP P4 implementation, P4 register accesses which affect control flow and lookups might not behave as expected. This is because the register values are not part of the cached lookup key. This will manifest itself as executed actions being repeated even though register values should have changed the actions to be executed.

  To mitigate this, a P4 pragma can be attached to an action definition that forces the cached lookup to be ignored. This pragma is as follows: `@pragma netro no_lookup_caching` This pragma should immediately precede the declaration of the action as referenced by the table action definition i.e. not attached to an action called by another action. Using this pragma will negatively impact performance. However, it will only degrade performance for flows where the action is executed. It is also possible that the ingress part of the flow can be cached while only the egress is bypassed.

Generally speaking, it may be possible to mitigate this effect using C plugin functions which allow conditional execution.

- Meters are also affected by interaction with cached lookups. It is possible to drop a packet marked `RED` automatically by attaching the following pragma to the meter declaration: `@pragma netro meter_drop_red` It is also possible to use the `no_lookup_caching` pragma to overcome this limitation. Though this will negatively impact performance.
- Register access is not atomic by default and it is possible to have many threads operating on registers simultaneously. This may lead to unexpected results. To overcome this, a pragma may be attached to a register declaration as follows: `@pragma netro reglocked`. This will create a mutex per register that grants exclusive access within all actions that reference that register.

3.3 Managed C Application Development

- Managed C application development is performed from Programmer Studio. A dataplane is generated as soon as the application is built with user code being invoked for each ingress packet.

Note
Managed C application development is experimental.

- User code assignment starts from ME i32:0. When additional worker MEs are assigned (from the Build Settings page) the additional MEs to be userd are allocated on subsequent islands. All MEs use the same list file `pif_app_nfd.list`.

Note
It is sometimes easier to debug an application when the number of worker MEs is set to a minimum as break points need to be set in fewer MEs.

Note
The minimum number of MEs that can be selected for hardware debugging is the number of ME islands in the selected FP-4xxx/NFP-6xxx variant while on the simulator it can be 1.

- During hardware debugging the RTE is used to configure the MACs and SR-IOV VFs. The debug server needs to be running on the target platform when debugging applications.

Note
The debug server can be started via upstart with `start nfp-hwdbg-srv`. It is then recommended to start the RTE in DEBUG MODE with `start nfp-sdk6-rte-debug` which will limit code to run on a single ME

- To enable VLAN offload set the global C define PKTIO_VLAN_ENABLED. The VLAN offload information is stored in the packet metadata field 'p_vlan. On ingress, the value will be non-zero when TX offload is used;

the field value indicates the VLAN ID in use. On egress, the value must be set to non-zero to indicate the RX VLAN ID offloaded. An API is introduced in "port_config.h" to retrieve the NFD port configuration, this can be used to check whether RX VLAN offloading is enabled for a given port.

Please refer to the Appendix in the *Development Tools Users Guide* for additional information.

3.4 Run Time Environment

- The RTE depends on and includes the latest released Board Support Package from the 6000-b0 branch (NFP-BSP-6000-B0). This version of the BSP now allows installation on supported operating systems with kernels newer than 4.10. Up to kernel 4.15 have been tested, alhtough not tested there are no known limitations running on kernels newer than 4.15.
- DEBUG_MODE is invoked via RTE input flag `--sdk_debug`

  All settings that is set with environmental variables have default values when not set.

4. Enhancements and fixes in NFP SDK 6.0.4.1

The 6.0.4.1 release includes enhancements and fixes listed in this section.

4.1 Run Time Environment

- Added support for Ubuntu 18.04 in the installation script.
- Fixed warning message about version mismatch.
- Corrected the updating of the initial RAM disk image in the CentOS installation script.
- Solved the issue of installing NFP BSP DKMS RPM package on CentOS 7.4 (3.10.0-693.17.1.el7.x86_64) and higher by implementing a workaround in the installation script.

5. Enhancements and fixes in NFP SDK 6.0.4

The 6.0.4 release includes enhancements and fixes listed in this section.

5.1 Programmer Studio Integrated Development Environment

- Fixed the bug that no dialog box popup when hitting a breakpoint while using the Simulator in Programmer Studio.
- Fixed Programmer Studio not opening saved project due to incorrect handling of script name in project JSON file causing script file name mismacthes.
- Fixed Programmer Studio becomming unresponsive when deleting all rules in a P4 user config file.
- Fixed Programmer Studio not closing simulator when running in Wine after ending a debug session.

5.2 P4 Programming Language

- Multicast now supports up to 64k ports.
- The allocation of packet and buffer credits now take into account the number of worker MEs. On a P4 wire, throughput improvements of 9% to 12% have been measured.

5.3 Run Time Environment

- P4 Table Rules not being applied consistently have been fixed.
- The RTE depends on and includes an updated Board Support Package which adds support for kernel 4.10 and possibly beyond.
- Initialisation of VFs have been improved and more than 8 VFs are now supported.

6. Enhancements and fixes in NFP SDK 6.0.3

The 6.0.3 release includes enhancements and fixes listed in this section.

6.1 Programmer Studio Integrated Development Environment

- When not all tables in a project have rules defined in the user config, there was an issue when switching the selection from a table with rules to one without the rules list was not updated. This has been fixed.
- ME performance statistics did not report ME stats for MEs running in 4 context mode.
- Fixed thread view not expanding nodes when a project was saved after debug started.
- Fixed a buid problem where the linker would report the application master list file being in use.
- A fix was added where a non-P4 project would result in an opening file error being reported.
- Fixed "No microengine instructions in list file" when building a P4 program when NFD is turned off.
- Fixed Managed C applications displaying the incorrect number of MEs in the build settings.

6.2 P4 Programming Language

- Added support for MAC synchronized timestamp, which is a 64-bit value composed of a 32-bit seconds value in the upper 32-bits and a nanosecond value in the lower 32-bits. Please refer to the `intrinsic_metadata.ingress_global_timestamp` field in the P4 Intrinsic Metadata described in the *Developer Tools Users Guide* for more details.
- Support for the matching of header fields with offsets greater than 256 bytes was added. The matching limit is now 1024 bytes into packet headers.
- Fixed rules not always being matched which only occured in the Centos version of the RTE.
- Adressed startup condition where packets could be processed before DCFL tables were configured.
- Fixed a lock not always being released for flows hashing to similar keys in the flow cache.
- Packets dropped before cloning could take place, are not cloned. If cloning takes place before the original packet is dropped, the cloned packet will continue on its path and the original packet will be dropped.
- A packet is cloned only once on a specific context, due to the egress_instance not cleared. Egress_instance is now cleared on each newly received packet.
- Fixed rule lookup not always matching when the flow cache was disabled.
- Fixed action duplication leading to bad action references in control flow.
- Fixed resubmitted packets being dropped when their egress_spec is not set. The packet is resubmitted regardless.
- Fixed shift bug which affected operands less than 32-bits.

6.2.1 C Sandbox

- For running a C Sandbox in the P4 dataplane please refer to the Appendix in the *Development Tools Users Guide* for more information.
- When changing a sandbox function name PS doesn't always detect the change and doesn't update the dataplane. When build errors are encountered doing a rebuild (ALT+F7) usually resolves it.

6.3 Managed C Application Development

- Support for VLAN offload was added.

6.4 Run Time Environment

- The DCFL host memory usage has been reduced.
- Fixed table breakpoints reporting errors in the RTE until new rules were added.
- ERR47 check has been added to the RTE initialization procedure, a path to alternative err47 check script can be specified using the --err47 input argument. If this argument is set to SKIP the ERR47 will not be done.

7. New features and enhancements in NFP SDK 6.0.2

The 6.0.2 release includes fixes and enhancements listed in this section.

7.1 Programmer Studio Integrated Development Environment

- The number of worker MEs running a P4 or Managed C application can be selected from the *P4/Managed C Build Settings* dialog. Depending on the NFP variant and whether building for the simulator or hardware the range is between 1 and 70. (When set to 1, the application will run on the master ME, i32:0).
- If an opened source file in Programmer Studio's editor contains a *#include* directive and you want to open the file, **Open Document** menu item has been added to the shortcut menu when user right click on the line that contains *#include* directive.

7.2 P4 Programming Language

- In this release a number of corner case issues in deparse have been resolved.
- Several P4 Parser Exception bugs have been fixed.
- It is now possible to call once off initalization functions for sandbox C code.
- Checks for control flow restricted actions have been added.
- Support for shift and modify_field_rng_uniform primitives were added.

7.3 Managed C Application Development

- Support for VLAN offload was added. To enable set the global C define PKTIO_VLAN_ENABLED. The VLAN offload information is stored in the packet metadata field 'p_vlan. On ingress, the value will be non-zero when TX offload is used; the field value indicates the VLAN ID in use. On egress, the value must be set to non-zero to indicate the RX VLAN ID offloaded. An API is introduced in "port_config.h" to retrieve the NFD port configuration, this can be used to check whether RX VLAN offloading is enabled for a given port.

Please refer to the Appendix in the *Development Tools Users Guide* for additional information.

7.4 Run Time Environment

- Loading and unloading time has been reduced by approximately a factor of four.
- NBI DMA drop is now disabled in default configurations. This fixes global reordering issues under load.

- A server crash relating to digests with multiple clients have been fixed.
- The RTE now initializes all ME pseudo-random generator seeds.
- Implemented a new rule storage system to improve RTE rule lookup and modification speed.
- Added support for RHEL in the CentOS installer script.
- The `pif_ctl_nfd_debug.sh` and `pif_ctl_nfd.sh` have been merged into a single script called `pif_ctl_nfd.sh`. DEBUG_MODE is now invoked via RTE input flag `--sdk_debug`

  All settings that is set with environmental variables will now have default values when not set.

8. New features and enhancements in NFP SDK 6.0.1

This release includes fixes and enhancements listed in this section.

8.1 Programmer Studio Integrated Development Environment

- When debugging a P4 application the **Toggle View** button (or Shift+F8) now switches from the P4 view to the C view and then to microcode view. From the microcode view it will switch back to the C view and then the P4 view.
- Programmer Studio now supports parallel build when it is enabled. Parallel build can be enabled using build settings, general page. For more information regarding parallel build please refer to the section, *General Build Settings* of the *Development Tools Users Guide*.
- Programmer Studio now supports multi ME breakpoints for P4 source files. For more information regarding multi breakpoints please refer to the section, *About Multi-Microengine Breakpoint Support* of the *Development Tools Users Guide*.

8.2 Managed C Application Development

- Debugging using the Windows simulator is supported from this release.

8.3 Run Time Environment

- Support for running of multiple SmartNICs on a single host has been added. This is done by starting an instance of RTE (and if debugging Hardware Debug Server) for each target SmartNIC. Please refer to Appendix D in *Development Tools Users Guide* or the README for detailed instructions.

Note
Multi-card support described in this section is only applicable to NFP SDK 6.0 and not any other of Netronome's software products.

- Support has been added for CentOS 7 and Ubuntu 16.04 host operating systems. Please note CentOS and Ubuntu version are released as separate tarballs.
- Systemd service files will now be installed on OS supporting systemd (CentOS 7 and Ubuntu 16.04).

  Support has now been added that RTE can be started and a SmartNIC loaded with a set of P4 design files, firmware and rules at system startup by either using the Upstart or systemd init files

  Please refer to the Appendix D in the *Development Tools Users Guide* or REAMDE for additional information.

- Added 2 features that can be used to speed up development, debugging and testing.

  Starting RTE with `--load_last_set` instead of specifying firmware, design and rules configuration file will cause the RTE to load the last loaded firmware. P4 design and configuration rules file. Please note that last loaded files is stored in temporary system locations and may be deleted as part of system cleanup if not used for couple of days/weeks.

  Starting RTE with `--skip_fw_load` instead of specifying firmware file will skip the lengthy firmware loading procedure. It is up to the user to still specify and load a compatible P4 design file to ensure no system errors occurs.
- The installation procedure has changed to be more user friendly, SmartNIC flash versions are now checked to determine if upgrade is necessary (eliminates unnecessary flashing), all MAC, TM and NBI configuration has been moved to the load script. NFP BSP will now only be upgraded if older version is detected, in this case the user will prompted if NFP BSP should be upgraded.

  The recommended installation process has changed to running the install script with `./sdk6_rte_install install` and is highly recommended to allow the upgrading of the NFP BSP when prompted.

Please refer to the Appendix D in the *Development Tools Users Guide* for additional information.

9. New features and enhancements in NFP SDK 6.0.0

This release includes new features and enhancements listed in this section.

9.1 Programmer Studio Integrated Development Environment

- Programmer Studio now supports P4, Managed C (experimental), Micro-C and Microcode application development. For P4 and Managed C applications a dataplane is constructed for basic packet transmission. The dataplane supports physical ports (on the Agilio-LX and CX SmartNICs) and SR-IOV VFs on the host.

> **Note**
> To create a P4 or a managed C mode application the appropriate checkbox must be selected when creating a new project.

> **Note**
> P4 Language specification 1.1 is selectable but in this release its use is still experimental.

> **Note**
> A host with an Agilio-LX or CX SmartNIC must support ARI (Alternative Routing ID) for SR-IOV VFs.

- The number of worker MEs running a P4 or Managed C application can now selected from the *P4/Managed C Build Settings* dialog. Depending on the NFP variant and whether building for the simulator or hardware the range is between 1 and 70. (When set to 1, the application will run on the master ME, i32:0).
- *P4 User Config* dialogs for P4 Parser Value Sets have been added.
- When debugging a P4 application the **Toggle View** button (or Shift+F8) now switches from the P4 view to the C view and then to microcode view. From the microcode view it will switch back to the C view and then the P4 view.
- Break points added on rules in P4 Table are now saved.

9.2 P4 Programming Language

- Language specification 1.0.2 is supported with the limitations listed in Section 3.2.2.

Note
P4 Language specification 1.1 is selectable but its use is still experimental.

- P4 Applications also support Micro-C code running in a C Sandbox. Sandbox code can be invoked from P4 actions and will execute in the context of the action code.

9.2.1 Language Constructs

This release adds support for the following P4 language constructs:

- Parsing
  - Parser value sets (eg:for MPLS labels)
  - Variable length headers
  - Checksum field list calculations
- Matching constructs
  - LPM128 matching
- Control flow constructs
  - Conditionals with fields greater than 32 bits
- Field lists and the following checksum and hash calculations:
  - csum16
  - xor16
  - crc16
  - crc32
- Action constructs
  - set_field_to_hash_index

9.2.2 C Sandbox

- For running a C Sandbox in the P4 dataplane please refer to the Appendix in the *Development Tools Users Guide* for more information.
- When changing a sandbox function name PS doesn't always detect the change and doesn't update the dataplane. When build errors are encountered doing a rebuild (ALT+F7) usually resolves it.

9.3 Managed C Application Development

- Managed C application development is performed from Programmer Studio. A dataplane is generated as soon as the application is built with user code being invoked for each ingress packet.

> **Note**
> Managed C application development is experimental.

- User code assignment starts from ME i32:0 (running `pif_app_nfd_master.list`). When additional worker MEs are assigned (from the Build Settings page) the additional MEs run `pif_app_nfd.list` and are allocated on subsequent islands.

> **Note**
> It is sometimes easier to debug an application when the number of worker MEs is set to a minimum as break points need to be set in fewer MEs.

> **Note**
> The minimum number of MEs that can be selected for hardware debugging is the number of ME islands in the selected FP-4xxx/NFP-6xxx variant while on the simulator it can be 1.

- During hardware debugging the RTE is used to configure the MACs and SR-IOV VFs. The debug server needs to be running on the target platform when debugging applications.

> **Note**
> The debug server can be started via upstart with `start nfp-hwdbg-srv`. It is then recommended to start the RTE in DEBUG MODE with `start nfp-sdk6-rte-debug` which will limit code to run on a single ME

- Debugging using the Windows simulator is supported in this release.

Please refer to the Appendix in the *Development Tools Users Guide* for additional information.

10. New features and enhancements in NFP SDK 6.0 Beta 1

This release includes new features and enhancements listed in this section.

10.1 Programmer Studio Integrated Development Environment

- Programmer Studio now supports P4, Mananged C (experimental), Micro-C and Microcode application development. For P4 and Managed C applications a dataplane is constructed for basic packet transmission. The dataplane supports physical ports (on the Agilio-LX and CX SmartNICs) and SR-IOV VFs on the host.

> **Note**
> Managed C application development is experimental.

- The number of worker MEs running a P4 or Managed C application can now selected from the *P4/Managed C Build Settings* dialog.
- *P4 User Config* dialogs for P4 Multicast, P4 Registers and P4 Meters have been added.
- When debugging a P4 application the **Toggle View** button (or Shift+F8) now switches from the P4 view to the C view and then to microcode view. From the microcode view it will switch back to the C view and then the P4 view.
- Break points can now be added on rules in P4 Tables. In this release the break points are however not persisted between debug sessions and will have to be set in each debug session.
- Programmer Studio now supports cross break point display in P4, C and microcode source view. This feature is automatically enabled for all new code break points inserted using SDK 6.0 Beta 1 (this release). Existing break points must be removed and re-inserted again to enable this feature.
- For P4 and Managed C applications NBI configuration will only list the MAC configurations with prepend enabled.

10.2 P4 Programming Language

- Language specification 1.0.2 is supported with the limitations listed Language Constructs section below.

> **Note**
> P4 Language specification 1.1 is selectable in this release but its use is still experimental.

- P4 Applications also support Micro-C code running in a C Sandbox. Sandbox code can be invoked from P4 actions and will execute in the context of the action code.

10.2.1 Language Constructs

This release adds support for the following P4 language constructs:

- Parsing
  - Parser handling exception packets
  - Select values greater than 32 bits
- Meters
- Action constructs
  - Register operations
  - Resubmit and recirculate
- Packet duplication
  - Multicast
  - Clone_ingress_pkt_to_ingress
  - Clone_ingress_pkt_to_egress
  - Clone_egress_pkt_to_ingress
  - Clone_egress_pkt_to_egress

Currently unsupported constructs are:

- Parsing
  - Variable length headers
  - Parser value sets (eg:for MPLS labels)
  - Checksum field list calculations
- Matching constructs
  - LPM128 matching
  - Arbitrary LPM matching
- Control flow constructs
  - Conditional with fields greater than 32 bits
- Action constructs
  - set_field_to_hash_index
  - Calculated fields

11. New features and enhancements in NFP SDK 6.0 Managed C Beta 1

Providing a Managed C environment is being considered for the 6 line of the SDK and the purpose of this release is to engage with users on the usability and desire for such a development model.

11.1 Programmer Studio Integrated Development Environment

- In this release Programmer Studio only supports developing Micro-C applications in a C Sandbox environment.

11.2 Network Flow Linker and Loader

- New option set, `-blob_*`, allows a user to insert an arbitrary file as a memory symbol at any location. The symbol is allocated as if it was declared with a fixed-offset directive and conflicts will be detected as errors. The name and alignment of this symbol can be set. See the linker help message for more details.
- New options `-usr_note, -usr_note_f`, allows a user to insert any text or file as an ELF note into the NFFW file. The linker and loader do not care about the contents of these notes and it does not affect the firmware. It can be used to bind extra build details to an NFFW file, to add application-level debug info or simply a description of the firmware.

12. New features and enhancements in NFP SDK 6.0 preview

The 6.0-preview release includes new features and enhancements listed in this section.

12.1 Programmer Studio Integrated Development Environment

- Programmer Studio now supports developing in the P4 programming language. (Refer to www.p4.org and the P4 Language section below)
- Various stability fixes.

12.2 P4 Programming Language

12.2.1 General limitations

It is possible that large P4 designs will hit either code store or various memory limitation within the NFP. The exact nature of these limitations is unknown at this time.

It is important to note that compound actions are executed in series, as opposed to the parallel as the 1.0 P4 specification dictates.

A number of P4 language constructs are not supported; these are listed below.

12.2.2 List of unsupported constructs

- Parsing constructs
  - Variable length headers
  - Parser value sets (eg:for MPLS labels)
  - Parser handling exception packets
  - Select values greater than 32 bits
  - Checksum field list calculations
- Matching constructs
  - LPM128 matching
  - Arbitrary LPM matching

- Control flow constructs
  - Conditional with fields greater than 32 bits
- Meters
- Action constructs
  - set_field_to_hash_index
  - calculated fields
  - register operations
  - resubmit and recirculate
  - Packet duplication
    - multicast
    - clone_ingress_pkt_to_ingress
    - clone_ingress_pkt_to_egress
    - clone_egress_pkt_to_ingress
    - clone_egress_pkt_to_egress

12.3 Network Flow Linker and Loader

- New option set, `-blob_*`, allows a user to insert an arbitrary file as a memory symbol at any location. The symbol is allocated as if it was declared with a fixed-offset directive and conflicts will be detected as errors. The name and alignment of this symbol can be set. See the linker help message for more details.
- New options `-usr_note, -usr_note_f`, allows a user to insert any text or file as an ELF note into the NFFW file. The linker and loader do not care about the contents of these notes and it does not affect the firmware. It can be used to bind extra build details to an NFFW file, to add application-level debug info or simply a description of the firmware.

13. New features and enhancements in NFP SDK 5.2.0

The 5.2.0 release includes new features and enhancements listed in this section.

13.1 General

This release of the SDK adds support for developing for the NFP-4xxx family of devices as well as for the B0 stepping of the NFP-6xxx family.

13.2 Programmer Studio Integrated Development Environment

- Support was added for watching user symbols in EMEM, IMEM, EMEM Data Cache, CTM, ILA SRAM, CRYPTO SRAM.
- When editing Micro-C files, a **Go To Function Definition** menu option is now availble when right-clicking on a function name. Ctrl+Shift+B can be used to return to the source line.
- Support was added for having break points on fields in Chip CSRs.
- Startup scripts will now also be executed when debugging on Hardware.
- Various stability fixes.

13.3 Network Flow Assembler

- Improvements to spilling including a new spilling mode `-spilling-no-code`.
- Improved warnings and illegal syntax detection.
- Optimization fixes, preventing illegal code generation on chips with errata.
- Additional memory bug fixes preventing crashes.

13.4 Network Flow C Compiler

- The compiler now supports "file-scope `__asm` blocks" which makes possible the convenient use, from Micro-C but using familiar NFP assembler syntax, of linker-oriented directives such as `.alloc_mem`, `.alloc_resource`, `.declare_resource`, `.init`, and `.init_csr`. A new compiler intrinsic `__link_sym` also allows ready access to linker symbols. In addition, it is now possible to specify codeless `.uc` files containing linker directives which will be processed by the NFP assembler and the output included in the resulting Micro-C list file.

- The compiler backend has been revised to provide a unified means for constant register initialization, both in the case of unspilled registers, and in the case of registers spilled to local and external memory.
- There is now fuller support for use, in "self" mode, of next-neighbor registers as local data storage, including subword access. Proper provision has also been made for the initialization of NN registers.
- Numerous improvements have been made to compiler diagnostics. For instance, an attempt to do subword assignments to transfer registers in now flagged as an error. A number of new warnings have been provided.
- Various fixes have been made to 40-bit address support, including making better provision for address size determination, and for the initialization of 64-bit pointers.
- A number of fixes have been implemented to code dealing with CLS reads and writes.
- Various fixes are in place to allow the appropriate use of local memory pointers 2 and 3.
- A large number of other fixes and incremental enhancements are in place.

13.5 Network Flow Linker and Loader

- The linker now adds Init-CSR entries to ensure that ctx0 is the first context to start and if there is an entry for ctx0.LMAddr (or related LMAddr CSRs), a matching entry for ActLMAddr is also added. This addresses an issue where one context might not have the correct LMAddr value on startup.
- Setting a memory resource size to 0 now works correctly. Previously it would be ignored (`-res.mem.size`)
- The `-f` option (fill codestore) has been restored and will now fill the entire codestore as expected. This is usually only useful for debugging purposes.
- Reduced debug data duplication for list files assigned to multiple microengines.
- New directives to pre-initialize memory unit queues or rings (`.load_mu_qdesc` and `.init_mu_ring`). The linker will generate microcode to load queue descriptors which is executed before loading user code. The first line of user code can therefore assume that the queue or ring is ready to use.
- Debug data for spilled registers in LMEM now correctly uses the context multiplier - watches from Programmer Studio will now show the data from the correct context.
- Added command line option `-name` to set the firmware name in the MIP. This helps identifying loaded firmware when using `nfp-nffw status`.
- A codeless list file can now be added to the link command. This list file can contain global-scope symbol definitions and initializations. See `nfas -codeless` and `nfld -L codeless.list`.
- Init-CSR directives now supports the same bash-style range formatting as nfp-reg. For example, `.init_csr mecsr:i{32..34}.me{0,4,8}.Mailbox{0..3} 0x00ffbea7`
- New predefined load-time import variable `__CHIP_REV` will be resolved to the revision of the chip the firmware gets loaded on. This currently needs to be placed in a register to be used. For example,

```
                            immed[r0, __CHIP_REV]
                            .if (r0 < __REVISION_B0)
                                // A0 specific code
                            .endif
```


- Added option `-sym_across_32bit` to control how the linker handles symbols that cross the 32-bit addressing boundary.
- New options `-emu_cache_as_ddr` and `-show_res_cfg` can be used to build common code for targets that may or may not have DDR populated. Rather than requiring the user to use emem_cache as symbol resource declarations, this option turns the ememX allocation pool into an alias for ememX_cache.
- New MIP version to support larger address ranges and rtsymtab and rtstrtab offsets. For this release, the default behaviour of `-mip` is `-mip_v1`. Use `-mip_v2` if you have a loader with MIP v2 support and need rtstrtab to be larger than 64KiB. A future release will make MIP v2 the default - it is capable of more features which may be used for various future features.
- New option `-pad_align` can be used to control separation of symbols by making sure they have a minimum alignment and size as specified. Note that the symbol size is always the size the user code specifies, but during symbol allocation the linker will allocate symbols with padding that will essentially be unused space.
- When list files are used in duplicate shared codestore mode, memory symbols with scope "me" would automatically become "tg" scope since both MEs now use the same code referring to the same symbol (excluding LMEM). The same behaviour is now applied to `.declare_resource` and `.alloc_resource` scopes, not just to `.alloc_mem`.

13.6 Network Flow Simulator

- Support for simulating B0 silicon revisions as well as the NFP-4xxx family of devices have been added.

14. New features and enhancements in NFP SDK 5.1.0

The 5.1.0 release includes new features, enhancements and changes listed in this section.

14.1 General

This release of the SDK provides provides major alignment of the SDK APIs for interacting with the Simulator with those provided by the Board Support Package (BSP). This enables developers to write and debug C code for interacting with the NFP from Programmer Studio or the command line prompt using C-script (cling) on Windows or the GNU/LLVM tool chains typically available on Linux in a simulated NFP-6xxx environment.

Please refer to the sections, *NFP CPP Interfaces APIs*, *NFP Flow Firmware APIs*, *NFP MAC APIs*, *NFP NBI TM APIs* and *NFP NBI DMA APIs* in the *Netronome NFP Board Support Package Programmer's Reference Manual* for using these APIs. A description of these APIs are also available in the *NFP-6xxx Simulator API Reference Manual* available from the *ToolView* panel in Programmer Studio.

14.2 Programmer Studio Integrated Development Environment

- This release supports interactive hardware debugging from Programmer Studio. The debug server software needs to be extracted and built on the target platform from the distribution source code tarball. Refer to the *NFP-6xxx Debug Server User's Guide* in the *doc* directory from the tarball for more information.
- The following debug windows have been added:
  - NBI TM Packet Descriptor Watch.
  - Performance Statistics.
- The EMEM Data Cache can now be watched during debug from the *Memory Watch* debug window.

Note
The Crypto memory and Interlaken Look-Aside (ILA) memory watches are only available when debugging on variants of the NFP-6xxx which has them.

- MAC initialization from Programmer Studio is now performed using the *nfp-macinit* tool from the BSP. This ensures consistent MAC configuration between the simulator and hardware.

14.3 Network Flow Assembler

- Added `__nfp_has_island()` pre-processor function for determining whether the targeted device has a specific island.

- Many minor fixes and enhancements were added overall including to `.init_csr` and `.alloc_resource` directives introduced in the 5.0.0 release.
- isnum() is deprecated and replaced by is_ct_const()
- Alternative mnemonics have been added to improve consistency and ease of use.

  The mnemonics and their alternatives are listed in the tables below.

Table 14.1. Cluster Local Scratch: Mnemonics and their alternatives

Mnemonic	Alternative mnemonic
test_and_compare_write	test_compare_write
tcam_lookup	tcam_lookup32
sub_sat	subsat
sub_imm_sat	subsat_imm
add_imm_sat	addsat_imm
put_offset	ring_put_offset
put	ring_put
add_tail	ring_add_to_tail_ptr
add_sat	addsat
add_imm_sat	addsat_imm
test_and_set_imm	test_set_imm
journal	ring_journal
test_and_clr	test_clr
test_and_set	test_set
test_and_clr_imm	test_clr_imm
tcam_lookup_byte	tcam_lookup8

Table 14.2. Memory Unit: Mnemonics and their alternatives

Mnemonic	Alternative mnemonic
add64_imm_sat	addsat64_imm
add64_sat	addsat64
add_imm_sat	addsat_imm
add_sat	addsat
compare_write	compare_write_or_incr
get_safe	get_freely
pop_safe	pop_freely
sub64_imm_sat	subsat64_imm
sub64_sat	subsat64
sub_imm_sat	subsat_imm

Mnemonic	Alternative mnemonic
sub_sat	subsat
test_and_add64_imm	test_add64_imm
test_and_add64_imm_sat	test_addsat64_imm
test_and_add64	test_add64
test_and_add64_sat	test_addsat64
test_and_add_imm	test_add_imm
test_and_add_imm_sat	test_addsat_imm
test_and_add	test_add
test_and_add_sat	test_addsat
test_and_clr_imm	test_clr_imm
test_and_clr	test_clr
test_and_compare_write	test_compare_write_or_incr
test_and_set_imm	test_set_imm
test_and_set	test_set
test_and_sub64_imm	test_sub64_imm
test_and_sub64_imm_sat	test_subsat64_imm
test_and_sub64	test_sub64
test_and_sub64_sat	test_subsat64
test_and_sub_imm	test_sub_imm
test_and_sub_imm_sat	test_subsat_imm
test_and_sub	test_sub
test_and_sub_sat	test_subsat
test_and_xor_imm	test_xor_imm
test_and_xor	test_xor

Table 14.3. PCIE: Mnemonics and their alternatives

Mnemonic	Alternative mnemonic
read_pcie	read_int
write_pcie	write_int

14.4 Network Flow C Compiler

14.4.1 New features

- Support for initializing CSRs have been added. Refer to section 4.9.1 in the *NFCC User's Guide* for more information.
- Support for memory allocation has been improved by the use of the `alloc_mem` directive. This simplifies sharing of variables between Micro-C and Microcode. Section 4.9.2 in the *NFCC User's Guide* provides more information.
- Hierarchical resources are now supported. Refer to section 4.9.3 in the *NFCC User's Guide* for more information.
- The `-WX` compiler option was added to turn warnings into errors.
- The command line compiler defaults to non-verbose mode, and can be controlled by the `-verbose` option.
- Section 7.5, *Queries and pitfalls*, was added to the *NFCC User's Guide* for avoiding common and unexpected coding problems.

14.4.2 Enhancements and Fixes

- A number of improvements have been made to emitted code associated with writing to and reading from CTM, IMEM and EMEM.
- The compiler handling of intrinsic functions has been considerably improved.
- Various improvements have been made to error reporting. In addition, unrecognized `__declspec` attributes are now an error.
- Corrections have been made to register spilling support.
- Some issues with `-Qliveinfo` have been fixed.
- An issue with opening shared files from within a VM has been fixed.

14.4.3 Other Changes

- -Qrevision_max should be set to 0xff or simply omitted due an incorrect comparison to the highest revision of the target chip.
- Only big-endian mode is supported from this release.

Note
Please contact Netronome technical support if little-endian mode is needed as it may not be reinstated in a future release if there is no demand for it.

- The live range output produced (when using the -Qliveinfo compiler option) for a register may be incorrect when a member of a struct that maps into a register is written to (or modified). The liverange of the register is indicated as being from before the first write or modify operation. A workaround is to assign a value to the register before operating on the member through the struct. The live range will then be indicated as starting from that assignment.
- The compiler supports atomic variables, as documented in section 4.3.9 of the *NFCC User's Guide*. This support is still experimental and while it should work well for common cases, there may be issues if it is used indiscriminately.
- Compiling mixed C/microcode applications (`-uc` option) should not be used in this release.
- There may be issues writing and reading individual subwords (C char and short types) to and from local and external memory. In general, this is best avoided on efficiency grounds anyway, however. For example,

```
volatile __declspec(cls) char msg[4];
void main(void)
{
unsigned x;

msg[0] = 'h';
msg[1] = 'i';
msg[2] = '!';
msg[3] = '\0';
/* ... */
}
```

generates eight CLS commands, whereas prepared word-oriented I/O generates one CLS write.

```
*(volatile __declspec(cls) unsigned *)msg = 0x68692100;
```

14.5 Network Flow Linker and Loader

- MIP can now be located above 256KiB in emem0. The default MIP will still be allocated first to put it as close to the start of emem0 as possible.
- The `-dump_res` command line option was added to dump resource usage to JSON file.
- When using relative ActLMAddr mode and init_csr for ctx0.IndLMAddr CSRs, the linker adds an entry for ActLMAddr with the same value to ensure that the start-up value is correct.
- Only constant values can be used for .init_csr values. Support is being worked on to use memory symbols as values to allow easier initialization of things like DMA or local memory pointers.

14.6 Network Flow Simulator

- Added PCIe endpoint simulation using a QEMU host. For more information contact Netronome support for assistance.
- Updated and added APIs to align with those available in the BSP.
- The following tools from the BSP are now included for use with the simulator:
  - `nfp-cpp`
  - `nfp-reg`
  - `nfp-xpb`
  - `nfp-mactool`
  - `nfp-tminit`
  - `nfp-hwinfo`
  - `nfp-macinit`
  - `nfp-mereg`
  - `nfp-power`
  - `nfp-tcache`
  - `nfp-rtsym`
  - `nfp-mem`
- Added functionality to support the new debug features available in Programmer Studio.

14.7 Standard Library

- The NFP offers a unified view of the DRAM and SRAM transfer registers. In this release the standard library has been updated to use the consolidated view with the transfer register types updated as in Table 14.4 below.

Table 14.4. Deprecated transfer registers names and their replacements

Deprecated type	New type
sram_read_reg	read_reg
sram_write_reg	write_reg
sram_read_write_reg	read_write_reg
dram_read_reg	read_reg
dram_write_reg	write_reg
dram_read_write_reg	read_write_reg

The compatibilty include in `components\standardlibrary\microc\include\nfp6000\nfp_compatibility_6000.h` ensures code portability is maintained.

- The names of CLS reflect intrinsics have changed as in Table 14.5 below.

Table 14.5. Deprecated CLS reflect intrinsic names and their replacements

Deprecated name	New name
cls_reflect_from_sig_src (_ind)	cls_reflect_write_sig_local (_ind)
cls_reflect_from_sig_dst (_ind)	cls_reflect_write_sig_remote (_ind)
cls_reflect_from_sig_both (_ind)	cls_reflect_write_sig_both (_ind)
cls_reflect_to_sig_src (_ind)	cls_reflect_read_sig_remote (_ind)
cls_reflect_to_sig_dst (_ind)	cls_reflect_read_sig_local (_ind)
cls_reflect_to_sig_both (_ind)	cls_reflect_read_sig_both (_ind)

The compatibilty include in `components\standardlibrary\microc\include\nfp6000\nfp_compatibility_6000.h` ensures code portability is maintained.

- The names of some CLS ring intrinsics have changed as in Table 14.6 below.

Table 14.6. Deprecated CLS ring intrinsic names and their replacements

Deprecated name	New name
cls_pop_safe_ring	cls_ring_pop_safe
cls_get_safe_ring	cls_ring_get_safe
cls_pop_ring	cls_ring_pop
cls_put_ring	cls_ring_put
cls_get_ring	cls_ring_get
cls_journal_ring	cls_ring_journal

The compatibilty include in `components\standardlibrary\microc\include\nfp6000\nfp_compatibility_6000.h` ensures code portability is maintained.

- `cls_ring_put_offset()` has been removed. `cls_ring_write_offset()` should be used instead.
- The range of immediate data that can be used in MEM immediate intrinsics have changed when NFP-32xx indirect ref mode is used. Only the bytemask can be used when overriding the immediate value, therefore the range is as follows: 0-0x7f for non-arithmetic immediate intrinsics (unsigned) and 0-0xff for arithmetic immediate operations (signed).
- Functions in LibC supporting memories not available in the NFP-6xxx have been removed like Global Scratch and Ustore.
- Global Scratch has been removed but can be replaced with either CLS intrinsics (local communication) or MEM intrinsics (global communication). For Global Scratch ring intrinsics use either CLS ring intrinsics (local communication) or MEM ring intrinsics (global communication). Polling rings have been deprecated in NFP-32xx

and is not supported in NFP-6xxx, rather use MEM work queue intrinsics. Please keep in mind that MEM ring (MEM queue engine) is only available on external memory (i24/emem0, i25/emem1, i26/emem2), refer to *NFP-6xxx Programmers Reference Manual*.

- Intrinsics for MSF and CAP has been removed.
- CAP interthread signaling can be replaced with Cluster Target intrinsic, `cluster_target_signal_me_ctx()`. CAP reflect intrinsics can be replaced with Cluster Target reflect intrinsics, and CAP CSR intrinsics can be replaced with Cluster Target XPB write and read (`cluster_target_xpb_write()` and `cluster_target_xpb_read()`).
- DRAM has been changed to MEM to include IMEM, EMEM and CTM. Memory can be declared as:
  - CTM: `__declspec(i34.ctm, addr40)`
  - EMEM: `__declspec(i24.emem, addr40)`
  - IMEM: `__declspec(i28.imem, addr40)`
- SRAM has been deprecated but can still be used through VQDR for backwards compatibility.

Note
All SRAM/VQDR instructions are deprecated in NFP-6xxx and programmers should not use them. SRAM/VQDR instructions are ONLY available for porting old code that used QDR memory.

- If 32-bit addressing is used for DRAM or SRAM, alternative IMB settings is needed to ensure the command reaches the correct island. See "Recommended Settings for NFP-32x code ported to NFP-6xxx" in the *NFAS Users Guide*. If 32-bit addressing is used but DRAM is declared with `__declspec(dram, ptr40)` and the intrinsic variant with `xx_ptr40` in the declaration is used, then the default IMB settings can be used.
- If 32-bit addressing with CLS and CTM is used; then access will be only to local island CLS and CTM.
- All CLS and MEM intrinsics have `xx_ptr32()` and `xx_ptr40()` variants. The `ptr32()` variant can be used when accessing local CLS or CTM. The `ptr40()` variant must be used when accessing EMEM, IMEM or other island CLS and CTM. If NFCC compiles with the default option of 40-bit addressing, then the CLS or MEM intrinsic of `xx()` is compiled to the variant `xx_ptr40()`. If the NFCC compilation option of `-compat32` is used then the CLS or MEM intrinsics is compiled to the variant `xx_ptr32()`.
- Support for dual DRAM signals in MEM bulk transfer intrinsics have been removed as it is not needed by the NFP-6xxx architecture.
- Miscellaneous intrinsics `__ME, __island` could be substituted with `__LoadTimeConstant()`. Refer to `__LoadTimeConstant()` under Miscellaneous Functions in *Micro-C standardlib Reference Manual*.

14.8 Network Flow Simulator

- The simulator access API has been changed to align the API provided by the BSP. For more information refer to the *NFP-6xxx Simulator API Reference Manual*.

Note
At the time of this release, the NFP-6xxx BSP API was still under development and the APIs provided is this release may change in the next SDK release.

- Stream files or PCAP files need to be created manually or created / captured using third party tools.
- A single NFP chip can be simulated at any point in time (i.e. multi-chip simulation is not supported).
- The following NFP features are not simulated:
  - ARM within the ARM island.
  - PLL.
- Parts of the PCIe, MAC and DDR3 controllers are based on functional models (rather than logic) and are not necessarily cycle accurate representations.

14.9 Simulation and Debugging

- 4.x script files (`.ind`) will need to be updated to execute correctly with the NFP-6xxx Simulator.
- When hardware debug encounters a soft break point, `ctx_arb[bpt]`, Programmer Studio will respond with the following message: "The break point hit reported by the target is not recognized". Workaround is not to manually add soft breakpoints to application code but rather use Programmer Studio to set breakpoints in this release.
- The step debug button may enter a state where it continually activates and deactivates after starting a hardware debug session. There is currently no workaround other than clicking the button when it is in the active state.

15. New features and enhancements in NFP SDK 5.0.0 Beta

The 5.0.0 Beta release includes new features and enhancements listed in this section.

15.1 General

- Microengine identification throughout the SDK will now follow a textual presentation of the form "iX.meY" to indicate island X, microengine Y within that island. Hexadecimal or decimal integers should not be used to identify a microengine. This is to avoid confusion encountered for NFP-32xx where an integer may either indicate a linearly numbered microengine or a cluster and microengine hardware ID in hexadecimal form. For NFP-6xxx, linear numbering is not possible. Island aliases provide a more readable island ID, for example "emem0" instead of "i24". For a full list, see the Linker section in the *Developer Tools Users Guide*.

15.2 Programmer Studio Integrated Development Environment

- The following debug windows have been added:
  - NBI Memory Watch.
  - NBI Packet Modification Pipeline.
- Support for watching EMEM, IMEM, CTM, Crypto and ILA SRAM during simulation have been added to the *Memory Watch* debug window.

Note
The crypto memory watch is only available when simulating with a crypto capable variant of the NFP-6xxx.

- The *Data Watch* debug window now offers an option to select Chip CSRs for viewing during a debug session.
- The *Packet Streaming Configuration* dialog in Programmer Studio now provides a selection of MAC configurations. When a configuration is selected, Programmer Studio will during debug startup, run a configuration script to perform the required MAC setup.

  The following MAC configurations can be selected for each NBI:
  - 12x10GE
  - 12xIL
  - 1x100GE
  - 3x40GE
  - 4xIL-2x40GE

  - 4xIL-8x10GE
  - 8xIL-1x40GE
  - 12x1GE
  - 1x100GE-2x10GE
  - 1x10GE
  - 4x10GE-2x40GE
  - 4xIL-4x10GE-1x40GE
  - 8x10GE-1x40GE
  - 8xIL-4x10GE
- When some islands are unused by an application their clocks can be turned off from the *Simulation Options→Enabled Islands* dialog to improve simulation speed.
- Remote simulators can be used by specifying the connection in the *Simulation Options→Connections* dialog.

Note
Due to the number of interactions between Programmer Studio and the simulator, the network latency between the two should be minimized and a wired LAN should be given preference.

15.3 Network Flow Assembler

- Directives for Hierarchical Memory and Resource Allocation has been added. This enables resources like for example rings or memory regions to be managed in a central way by the linker opening the way to simpler and cleaner code. More information on the `.declare_resource` and `.alloc_resource` directives can be obtained in the *NFAS Users Guide*.
- The `.alloc_mem` directive is introduced to replace the `.local_mem` and `.global_mem` directives to perform memory allocation. It allows the scope of the allocation to be specified and preference should be given to its use in new code.

  Along with this directive, trailing attributes may be added to define the bit width of the address the symbol may assume. These are `addr32` and `addr40` and provide users with a means of validating the final address allocated and resolved by the linker. This is important when considering the address translations performed in the NFP-6xxx for which the linker will set destination island bits accordingly in the upper bits of an address. The default is addr40 for all memory types in the assembler.

  The `reserved` attribute introduced in the previous NFP SDK still behaves as it did before and may be useful in NFP-6xxx for reserving parts of CTM for packet data if other parts will be used for memory symbols allocated by the linker.
- The `.init_csr` directive was added to specify load time CSR initializations. Refer to *Init-CSR* in the *NFAS Users Guide*. Init-CSR acts as a contract between firmware and load target, letting the firmware state read-only requirements or writable CSR initializations which the loader will validate against the target's CSR database. If

there are conflicting requirements, the firmware will not be loaded. It can also be used in code libraries to state assumptions or requirements on certain chip or ME settings.

- A directive for link time assertions was added. It will cause linking to terminate when the expression passed to `.assert` evaluates to false. If the expression can already be evaluated at assembly time, the assembler will report the assert if it evaluates to false.
- Support was added for the new instruction mnemonics available in the NFP-6xxx. Refer to the *NFP-6xxx Programmers Reference Manual* for more details.
- The predefined macro, `__NFP_INDIRECT_REF_FORMAT_NFP6000`, is now available when using the MEv2.8 native indirect format.
- New built-in functions, `__nfp_meid` and `__nfp_idstr2meid` provide a way of identifying microengines in source code for functions such as `&remote`. For NFP-32xx, hexadecimal hardware microengine IDs would be used here. For NFP-6xxx, these new functions should be used. The integer produced by these functions is a hardware master ID.

15.4 Network Flow C Compiler

- Compiler support for declaring and operating on variables in 40-bit memory was added to enable access to the full 8G DRAM available in the EMEM islands in the NFP. 40-bit types are declared with `__declspec(i24.emem, addr40)` type qualifier. Please refer to the *NFCC User's Guide* for more information.
- Support was added for declaring and defining generic memory unit pointers using the `__declspec(mem_ptr)` qualifier.
- The compiler now uses 40-bit addressing by default, The `-compat32` option can be used for 32-bit addressing.
- Support was added for specifying 40-bit or 32-bit addressing using the `__declspec(addr32)` and `__declspec(addr40)` qualifiers.
- The maximum alignment that may be specified when referring to any external memory has been increased from 2K to 64M.
- Support for the NFP-6xxx indirect reference mode was added through the `-indirect_ref_format_nfp6000` option. The NFP-32xx indirect ref mode can still be selected using the `-indirect_ref_format_nfp3200` option.
- `__is_in_emem()`, `__is_in_imem()`, `__is_in_ctm()` was added to support the EMEM, CTM, IMEM memory types.
- Support for waiting on single signals in signal pairs is now possible by using the new `__wait_for_any_single()` and `__wait_for_all_single()` intrinsics.
- Similar to the assembler, new intrinsic functions have been added for producing MEIDs: `__nfp_meid` and `__nfp_idstr2meid`.
- Other new intrinsics include `__associate_read_write_reg_pair_no_spill()`, `__is_nfp_arch()` and `__is_nfp_arch_or_above()`, `__is_read_reg()`, `__is_write_reg()`, and `__is_xfer_reg()`.
- Various longstanding optimizer defects have been resolved.

15.5 Standard Library

- The Micro-C intrinsic library has been extended to cover the new functionality available in the NFP-6xxx. This includes intrinsics for:
  - Load Balance Engine
  - Lookup Engine
  - Packet Engine
  - Interlaken LA
  - NBI
  - Statistics Engine
  - Cluster Target
- Support for the Indirect Reference format of the MEv2.8 has been added.
- Support for CLS ring initialization has been added with intrinsic `cls_ring_init()`.

15.6 Network Flow Linker

- An updated ELF file format is used for NFP-6xxx with a new `EM_NFP6000` machine type. This accommodates the new Init-CSR mechanisms and changes the loading process to a more flexible process.

  For NFP-6xxx, only ELF64 output files are produced.

  The addresses of the symbols in the ELF symtab and in the linker produced .map file do not include island ID destination bits which the IMB CPP Address Translation needs. The microengine instructions using memory symbols will have translated/routable addresses, but the symbol table shows a global, untranslated, view.
- The linker will perform Init-CSR validation between list files as well as its own internal database of CSRs it needs to control. Init-CSR entries in list files are all considered to be part of the same database, so `const` and `required` entries merge into `const` for example, while this is not a valid merge between two CSR databases.
- All microengine registers (GPR, Xfer, NN) are now initialized using the Init-CSR mechanism instead of executing code on hardware or embedding the initializations in debug data for simulation. While this may not be noticeable, it means the same initialization sections in the NFFW file will be used for both hardware and simulation.
- Several new pre-defined import variables are available and listed in the *NFAS Users Guide*, such as `__MEID`, `__ISLAND`, `__MENUM`, `__ADDR_EMEM0`, and `__ADDR_I8_NBI`.
- The linker now uses resource pools to handle memory and generic resource allocation. Memory pools are defined in nfp_chipdata and interpreted by the linker based on where a list file is assigned. The simple `cls` pool, for example, will resolve to the CLS memory resource of the island the list file is assigned to. The `emem` resource pool, for example, allocates symbols from all available EMEM islands in a spread fashion, using the next island for each symbol until it finds space. Most resource pools map directly to only one memory resource. These resource pools are used to define hierarchical generic resources declared with `.declare_resource`

15.7 Network Flow Simulator

- The simulator architecture has been changed to a client server model. On a default install the simulator is run on the same machine as Programmer Studio but for scenarios where a more powerful server machine is available the simulator can be run on the server.

> **Note**
> Due to the number of interactions between Programmer Studio and the simulator, the network latency between the two should be minimized and a wired LAN should be given preference.

- The simulator now supports simulating the three EMEM islands each with 4GB or 8GB DRAM, single or dual channel with leveling. The full memory range is accessible from the SIM API as well as Programmer Studio.
- The Packet Processing Cores (PICO engines) are fully simulated. Their firmware can be selected from the *Linker* Tab in the *Build Settings* dialog in Programmer Studio.
- History collection has been extended to cover the new functionality and features available in the NFP-6xxx. Support for collecting DDR history for the 3 external memory units has also been added.

15.8 Scripting

- The C script interpreter from 4.x releases has been replaced by a more powerful and standards compliant interpreter. For more information, refer to the *cling - C interpreter* chapter in the *Development Tools User's Guide*.

> **Note**
> Due to the changes in the Simulator API and C interpreter, 4.x `.ind` files will have to updated to function on 5.x.

- Scripts can be executed from Programmer Studio from the *FileView* from the Project Workspace, *Simulation Options→Startup* and *View→Debug Windows→Command Line*.
- C commands can be interactively issued from the *View→Debug Windows→Command Line* window.

Appendix A. SmartNIC Product Numbers

Note

Note: Additional product numbers may appear in the SDK but are for internal use.

Table A.1. SmartNIC Product Numbers

Product Number	Agilio	Description	Network Ports	Memory	Crypto
AMDA0058-0011	LX	Starfighter Active Heatsink	2x40 GbE	8GB	Yes
AMDA0058-0012	LX	Starfighter Passive Heatsink	2x40 GbE	8GB	Yes
AMDA0078-0011	LX	Starfighter Active Heatsink	1x100 GbE	8GB	Yes
AMDA0078-0012	LX	Starfighter Passive Heatsink	1x100 GbE	8GB	Yes
AMDA0081-0001	CX	Hydrogen	1x40 GbE	2GB	No
AMDA0096-0001	CX	Lithium	2x10 GbE	2GB	No
AMDA0097-0001	CX	Beryllium	2x40 GbE	2GB	No
AMDA0099-0001	CX	Carbon	2x25 GbE	2GB	No

16. Technical Support

To obtain additional information, or to provide feedback, please email <support@netronome.com> or contact the nearest **Netronome** technical support representative.