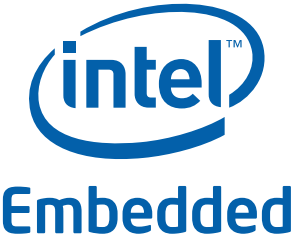

Intel® Data Plane Development Kit - Command Line Sample Application

User Guide

April 2012

Intel Confidential

Reference Number: 482246-1.1

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

A "Mission Critical Application" is any application in which failure of the Intel Product could result, directly or indirectly, in personal injury or death. SHOULD YOU PURCHASE OR USE INTEL'S PRODUCTS FOR ANY SUCH MISSION CRITICAL APPLICATION, YOU SHALL INDEMNIFY AND HOLD INTEL AND ITS SUBSIDIARIES, SUBCONTRACTORS AND AFFILIATES, AND THE DIRECTORS, OFFICERS, AND EMPLOYEES OF EACH, HARMLESS AGAINST ALL CLAIMS COSTS, DAMAGES, AND EXPENSES AND REASONABLE ATTORNEYS' FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PRODUCT LIABILITY, PERSONAL INJURY, OR DEATH ARISING IN ANY WAY OUT OF SUCH MISSION CRITICAL APPLICATION, WHETHER OR NOT INTEL OR ITS SUBCONTRACTOR WAS NEGLIGENT IN THE DESIGN, MANUFACTURE, OR WARNING OF THE INTEL PRODUCT OR ANY OF ITS PARTS.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to: http://www.intel.com/design/literature.htm.

Any software source code reprinted in this document is furnished for informational purposes only and may only be used or copied and no license, express or implied, by estoppel or otherwise, to any of the reprinted source code is granted by this document.

Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.

Copyright © 2012, Intel Corporation. All rights reserved.

Contents

Revision History

Date	Revision	Description
April 2012	1.1	Updates for software release 1.2
September 2011	1.0	Initial release

1.0 Introduction

This document describes the Command Line sample application that is part of the Intel® Data Plane Development Kit (Intel® DPDK).

1.1 Documentation Roadmap

The following is a list of Intel® DPDK documents in suggested reading order:

- **Release Notes**: Provides release-specific information, including supported features, limitations, fixed issues, known issues and so on. Also, provides the answers to frequently asked questions in FAQ format.
- **Getting Started Guide**: Describes how to install and configure the Intel® DPDK software; designed to get users up and running quickly with the software.
- **Programmer's Guide**: Describes:
  - The software architecture and how to use it (through examples), specifically in a Linux* application (linuxapp) environment
  - The content of the Intel® DPDK, the build system (including the commands that can be used in the root Intel® DPDK Makefile to build the development kit and an application) and guidelines for porting an application
  - Optimizations used in the software and those that should be considered for new development

  A glossary of terms is also provided.
- **API Reference**: Provides detailed information about Intel® DPDK functions, data structures and other programming constructs.
- **Sample Application User Guides**: A set of guides, each describing a sample application that showcases specific functionality, together with instructions on how to compile, run and use the sample application.

2.0 Overview

The Command Line sample application is a simple application that demonstrates the use of the command line interface in the Intel® DPDK. This application is a readline-like interface that can be used to debug an Intel® DPDK application, in a Linux* application environment.

Caution: The `rte_cmdline` library should not be used in production code since it is not validated to the same standard as other Intel® DPDK libraries. See also the "rte_cmdline library should not be used in production code due to limited testing" item in the "Known Issues" section of the Release Notes.

The Command Line sample application supports some of the features of the `GNU readline` library such as, completion, cut/paste and some other special bindings that make configuration and debug faster and easier.

The application shows how the `rte_cmdline` application can be extended to handle a list of objects. There are three simple commands:

- `add obj_name IP`: Add a new object with an IP/IPv6 address associated to it.
- `del obj_name`: Delete the specified object.
- `show obj_name`: Show the IP associated with the specified object.

Note: To terminate the application, use **Ctrl**-**D**.

3.0 Compiling the Application

1. Go to example directory:

```
export RTE_SDK=/path/to/rte_sdk
cd ${RTE_SDK}/examples/cmdline
```

2. Set the target (a default target is used if not specified). For example:

```
export RTE_TARGET=x86_64-default-linuxapp-gcc
```

   Refer to the *Intel® DPDK Getting Started Guide* for possible `RTE_TARGET` values.

3. Build the application:

```
make
```

4.0 Running the Application

To run the application in linuxapp environment, issue the following command:

```
$ ./build/cmdline -c f -n 4
```

Refer to the *Intel® DPDK Getting Started Guide* for general information on running applications and the Environment Abstraction Layer (EAL) options.

5.0 Explanation

The following sections provide some explanation of the code.

5.1 EAL Initialization and cmdline Start

The first task is the initialization of the Environment Abstraction Layer (EAL). This is achieved as follows:

```
int
    MAIN(int argc, char **argv)
    {
        ret = rte_eal_init(argc, argv);
        if (ret < 0)
            rte_panic("Cannot init EAL\n");
```

Then, a new command line object is created and started to interact with the user through the console:

```
    cl = cmdline_stdin_new(main_ctx, "example> ");
    cmdline_interact(cl);
    cmdline_stdin_exit(cl);
```

The `cmdline_interact()` function returns when the user types **Ctrl-d** and in this case, the application exits.

5.2 Defining a cmdline Context

A cmdline context is a list of commands that are listed in a NULL-terminated table, for example:

```
cmdline_parse_ctx_t main_ctx[] = {
    (cmdline_parse_inst_t *)&cmd_obj_del_show,
    (cmdline_parse_inst_t *)&cmd_obj_add,
    (cmdline_parse_inst_t *)&cmd_help,
    NULL,
};
```

Each command (of type cmdline_parse_inst_t) is defined statically. It contains a pointer to a callback function that is executed when the command is parsed, an opaque pointer, a help string and a list of tokens in a NULL-terminated table.

The rte_cmdline application provides a list of pre-defined token types:

- String Token: Match a static string, a list of static strings or any string.
- Number Token: Match a number that can be signed or unsigned, from 8-bit to 32-bit.
- IP Address Token: Match an IPv4 or IPv6 address or network.
- Ethernet* Address Token: Match a MAC address.

In this example, a new token type obj_list is defined and implemented in the parse_obj_list.c and parse_obj_list.h files.

For example, the cmd_obj_del_show command is defined as shown below:

```
struct cmd_obj_add_result {
    cmdline_fixed_string_t action;
    struct object *obj;
};


static void cmd_obj_del_show_parsed(void *parsed_result,
                    struct cmdline *cl,
                    __attribute__((unused)) void *data)
{
    /* ...*/
}


cmdline_parse_token_string_t cmd_obj_action =
    TOKEN_STRING_INITIALIZER(struct cmd_obj_del_show_result,
                    action, "show#del");

parse_token_obj_list_t cmd_obj_obj =
    TOKEN_OBJ_LIST_INITIALIZER(struct cmd_obj_del_show_result, obj,
                    &global_obj_list);

cmdline_parse_inst_t cmd_obj_del_show = {
    .f = cmd_obj_del_show_parsed,  /* function to call */
    .data = NULL,      /* 2nd arg of func */
    .help_str = "Show/del an object",
    .tokens = {        /* token list, NULL terminated */
        (void *)&cmd_obj_action,
        (void *)&cmd_obj_obj,
        NULL,
        },
    };
```

This command is composed of two tokens:

- The first token is a string token that can be `show` or `del`.
- The second token is an object that was previously added using the `add` command in the `global_obj_list` variable.

Once the command is parsed, the `rte_cmdline` application fills a `cmd_obj_del_show_result` structure. A pointer to this structure is given as an argument to the callback function and can be used in the body of this function.

§ §