OpenPOWER CAPI 2.0
Accelerator Compliance
Test Specification
Workgroup Specification
Revision 1.0 (January 30, 2018)
Abstract

The purpose of the OpenPOWER CAPI Accelerator Compliance - Test Specification is to provide the test suite requirements to demonstrate compliance of an OpenPOWER CAPI Accelerator solution with the OpenPOWER Power Service Layer (PSL) to Accelerator Function Unit (AFU) Interface Specification and the OpenPOWER Coherent Accelerator Interface Architecture Version 1 (CAIA V1) Specification.

This document is a Standard Track, Work Group Specification work product owned by the Compliance Workgroup and handled in compliance with the requirements outlined in the OpenPOWER Foundation Work Group (WG) Process document. It was created using the Master Template Guide version 1.0.0. Comments, questions, etc. can be submitted to the public mailing list for this document at <capi_accelerator@mailinglist.openpowerfoundation.org>.
Table of Contents

Preface .......................................................................................................................................... iv
  1. Conventions ............................................................................................................................. iv
  2. Document change history ......................................................................................................... v
1. Introduction ................................................................................................................................ 1
  1.1. Conformance to this Specification .................................................................................... 2
2. PSL-AFU Test Harness and Test Suite ...................................................................................... 3
  2.1. Test Harness to Execute the PSL-AFU Test Suite ............................................................... 3
  2.2. PSL-AFU Test Suite Required Tests ................................................................................... 3
  2.3. PSL-AFU Test Suite Optional Facilities Tests ................................................................. 4
  2.4. Successful Execution of PSL-AFU Required Tests ............................................................ 5
  2.5. Successful Execution of PSL-AFU Optional Facilities Tests ............................................. 5
3. CAIA Test Harness and Test Suite ............................................................................................ 6
  3.1. Test Harness to execute the CAIA Test Suite .................................................................. 6
  3.2. CAIA Test Suite Required Tests ...................................................................................... 6
  3.3. CAIA Hardware Optional Facilities Tests ......................................................................... 7
  3.4. Successful Execution of CAIA Required Tests ................................................................. 7
  3.5. Successful Execution of CAIA Optional Facilities Tests ................................................... 7
A. OpenPOWER Foundation overview .......................................................................................... 8
  A.1. Foundation documentation .............................................................................................. 8
  A.2. Technical resources ........................................................................................................ 8
  A.3. Contact the foundation .................................................................................................... 9
Preface

1. Conventions

The OpenPOWER Foundation documentation uses several typesetting conventions.

Notices

Notices take these forms:

- **Note**
  A handy tip or reminder.

- **Important**
  Something you must be aware of before proceeding.

- **Warning**
  Critical information about the risk of data loss or security issues.

Changes

At certain points in the document lifecycle, knowing what changed in a document is important. In these situations, the following conventions will used.

- **New text will appear like this.** Text marked in this way is completely new.

- **Deleted text will appear like this.** Text marked in this way was removed from the previous version and will not appear in the final, published document.

- **Changed text will appear like this.** Text marked in this way appeared in previous versions but has been modified.

Command prompts

In general, examples use commands from the Linux operating system. Many of these are also common with Mac OS, but may differ greatly from the Windows operating system equivalents.

For the Linux-based commands referenced, the following conventions will be followed:

**$ prompt** Any user, including the root user, can run commands that are prefixed with the $ prompt.

**# prompt** The root user must run commands that are prefixed with the # prompt. You can also prefix these commands with the sudo command, if available, to run them.
Document links

Document links frequently appear throughout the documents. Generally, these links include a text for the link, followed by a page number in parenthesis. For example, this link, Preface [iv], references the Preface chapter on page iv.

2. Document change history

This version of the guide replaces and obsoletes all earlier versions.

The following table describes the most recent changes:

<table>
<thead>
<tr>
<th>Revision Date</th>
<th>Summary of Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>January 30, 2018</td>
<td>• CAPI 2.0 Rev 1.0 WG Specification</td>
</tr>
<tr>
<td>October 25, 2017</td>
<td>• CAPI 2.0 Rev 1.0 - Public Review Draft</td>
</tr>
<tr>
<td>September 25, 2017</td>
<td>• CAPI 2.0 Rev 1.0 - pre2 - Updates from internal review</td>
</tr>
<tr>
<td></td>
<td>Added clarification that PSLSE demo kit pointer is still CAPI 1.0 but applicable for learning for CAPI 2.0.</td>
</tr>
<tr>
<td>August 22, 2017</td>
<td>• CAPI 2.0 Rev 1.0 - Initial updates for CAPI2.0</td>
</tr>
</tbody>
</table>
1. Introduction

The purpose of the OpenPOWER CAPI 2.0 Accelerator Compliance Test Harness and Test Suite (TH/TS) Specification is to provide the test suite requirements to be able to demonstrate OpenPOWER CAPI 2.0 Accelerator compliance. It contains the following:

- Section describing the test harness needed to execute the test suite
- Section describing the tests required to be in the test suite
- Section describing the successful execution of the test suite, including what it means for an optional feature to fail

The initial version of this document will be based on the POWER9 systems. It is expected that this document shall be updated for next generation OpenPOWER systems.

The input to this specification are the following two specifications:

1. OpenPOWER Coherent Accelerator Interface Architecture Version 2 (CAIA V2) Specification which describes a coherent accelerator interface structure for coherently attaching accelerators to the POWER9 systems using a standard PCIe bus

2. OpenPOWER Power Service Layer (PSL) to Accelerator Function Unit (AFU) Interface CAPI 2.0 Specification, Revision 1.0, which describes the interface to communicate to the acceleration logic running on the FPGA

These two specifications define the hardware interfaces as shown in the following figure which are the subject of this OpenPOWER CAPI 2.0 Accelerator Compliance document.

![Diagram of hardware interfaces](image)

Note on Endianness: The PSL is only a data mover across the interface, and there it has no concept of Endianness. It is up to the consumer of the data to interpret the Endianness of the data. The application and the accelerator determine the endianness of the data - the PSL just moves the data.
1.1. Conformance to this Specification

The following lists a set of numbered conformance clauses to which any implementation of this specification must adhere in order to claim conformance to this specification (or any optional portion thereof):

1. The required tests in the PSL-AFU Test Suite Required Tests Section and the CAIA Test Suite Required Tests Section must be successfully executed.

2. For optional facilities that are implemented, the optional tests in the PSL-AFU Test Suite Optional Tests Section and the CAIA Test Suite Optional Tests Section must be successfully executed.
2. PSL-AFU Test Harness and Test Suite

The purpose of this chapter is to provide the test suite requirements to be able to demonstrate OpenPOWER CAPI Accelerator compliance. Since problems found after hardware is built take significantly longer to fix compared to problems found prior to hardware being built, it is recommended for these tests to be executed and fixed pre-silicon. Then, run the tests again post-silicon.

- Section describing the test harness needed to execute the test suite
- Section describing the tests required to be in the test suite
- Section describing the successful execution of the test suite, including what it means for an optional feature to fail

2.1. Test Harness to Execute the PSL-AFU Test Suite

The Test Harness for verifying an AFU is a simulation environment. This simulation environment must emulate the PSL hardware behavior as it reacts to and drives the PSL-AFU Interface. The PSL Simulation Engine mentioned in this document is one example that is available right now. Others could become available in the future.

The PSL Simulation Engine (PSLSE) for CAPI 2.0 verifies that an AFU adheres to the PSL-AFU Interface CAPI 2.0 Specification. The PSLSE is available via GITHUB: https://github.com/ibm-capi/pslse/tree/capi2. The specific tests that are run are determined solely by what the Application and AFU are programmed to do. The Test Harness will verify that the PSL-AFU Interface CAPI 2.0 Specification is not violated.

Note

There is an example demonstration kit that includes a simple AFU available on IBM developerWorks at this link:  https://www.ibm.com/developerworks/community/forums/html/topic?id=be9384f5-1a5f-4654-8341-ed7d102ecd0c

The demonstration kit includes instructions on running PSLSE that will help explain how PSLSE works that is applicable to CAPI 2.0.

General recommendations for verification with PSLSE

1. PSLSE provides some randomness on the interface. Long running testcases or several runs of the same testcase will provide the most coverage.

2. Ensure the pslse.parms file is set correctly to control how PSLSE interacts with the AFU

2.2. PSL-AFU Test Suite Required Tests

The PSL Simulation Engine or any other test harness should test the requirements of an AFU:
1. AFU MMIO Interface
   A. AFU descriptor space is decoded correctly and valid values are returned. The Operating System requires that the AFU responds to valid data before it is enabled with a Start Command.

2. AFU Buffer Interface
   A. Must support either ah_brlat of 0, 1 or 2

3. PSL Response Interface
   A. Done
   B. AErr (Address Error – out of range or can’t be translated). Handle without hanging (driving jdone with jerror is ok).
   C. Derr (Data Error - internal PSL error). Handle without hanging (driving jdone with jerror is ok).
   D. Failed (Failed interrupt or bad context. Handle without hanging (driving jdone with jerror is ok).
   E. Paged is not required to be supported at this time with Linux
      i. in PSLSE set PAGED_PERCENT=0 in pslse.parms

4. AFU Control Interface
   A. Proper decode of a Reset Command with a jdone driven to indicate completion of Reset.
   B. Proper decode of a Start Command and assert jrunning until job is finished which is signaled by jdone.

2.3. PSL-AFU Test Suite Optional Facilities

Tests

1. Parity
   A. Parity checking on all interfaces is optional. The afu does not have to drive correct parity on the interfaces unless the AFU is driving ah_paren to ‘1’. This should be a static signal from the AFU though – either the AFU always wants parity enabled or always disabled.

2. AFU MMIO Interface
   A. It is optional for an AFU to have internal MMIO space (this is different from AFU descriptor space which is required). MMIO space support is reported in the AFU descriptor, and while this is optional it is suggested so that an AFU can capture errors or status information that can be read out by an application.

3. Command Interface
   A. AFU determines which commands it will send to the PSL. PSLSE will just verify that use of those commands do not violate the interface specification
B. Translation modes on ah_cabt are determined by the AFU design.
   i. The recommended modes are Strict or Paged.

4. Response Interface
   A. Context response support only required in AFU-Directed mode.
   B. Fault response support only if ah_cabt mode ABORT is used.
   C. COMP_EQ, COMP_NEQ and COMP_INV response support only required if the AFU sends CAS commands.

5. AFU Control Interface
   A. The ah_tbreq signal can be driven to ‘0’ if timebase is not required by the AFU.
   B. The ah_jcack signal is only used in AFU-Directed mode. In Dedicated mode this should be driven to ‘0’.
   C. The afu will not see a Timebase command if it doesn’t support sending a timebase .request (never asserts ah_tbreq).
   D. The afu will not see an LLCMD command encode unless it operates in AFU-Directed mode.
   E. The afu will not see an ASB_Notify Response encode unless it operates it sends an ASBNOT command.

6. DMA Interface
   A. The AFU is not required to use the DMA interface if it does not send DMA commands

2.4. Successful Execution of PSL-AFU Required Tests

The PSL Simulation Engine or any other test harness should indicate a Return Code 0 for successful completion with no errors. It should print error messages indicating what failed if there is a non-zero return code. If there is a non-zero return code the design should be debugged and fixed so that the tests can be run again.

2.5. Successful Execution of PSL-AFU Optional Facilities Tests

The PSL Simulation Engine or any other test harness should indicate a Return Code 0 for successful completion with no errors. It should print error messages indicating what failed if there is a non-zero return code. If there is a non-zero return code the design should be debugged and fixed so that the tests can be run again.
3. CAIA Test Harness and Test Suite

The CAIA defines hardware facilities and the required system software procedures for managing a compliant device. The CAIA Test Harness and Test Suite are defined for testing that the hardware facilities exist to allow this system software to manage the device. It is not intended to test the system software, libraries or applications.

3.1. Test Harness to execute the CAIA Test Suite

The CAIA Test Harness is an OpenPOWER Ready system that is CAPI enabled. The purpose of the Test Harness is to provide an environment to test for the existence of required CAIA facilities and implemented optional CAIA facilities within the CAPI hardware components of the system. The test harness must have a privileged mode application that uses the Open Source CAPI Kernel Driver or equivalent and uses the CAPI Accelerator Library (libcxl) or equivalent, and must have a matching test harness AFU that responds to operations as specified by the PSL-AFU interface.

There are two modes described in the CAIA spec: 1) directed mode, and 2) dedicated mode. A test harness is needed for each of the supported modes that claim compliance.

Note: The PSL (Power Service Layer) is defined in the CAIA spec generically. Any hardware that provides CAIA compliance is called a PSL, and it is that function that is being tested.

3.2. CAIA Test Suite Required Tests

The CAIA Test Suite required tests are generated by writing a functional application that tests the existence of the required CAIA facilities and verifies the correct behavior. The Open Source CAPI Kernel Driver or equivalent can be used to get access to privileged CAIA facilities, and the CAPI Accelerator Library (libcxl) or equivalent can be used to get access to user-mode CAIA facilities. Tests are needed in the test suite for each of the supported modes (directed mode or dedicated mode) that claim compliance.

The following are listed in the CAIA as required hardware facilities.

- PSL State Register (PSL_SR_An)
- PSL ID Register (PSL_ID_An)
- PSL Scheduled Process Area Pointer Register (PSL_SPAP_An)
- PSL Linked List Command Register (PSL_LLCMD_An)
- PSL Slice Control Register (PSL_SCNTL_An)
- PSL IVTE Limit Register (PSL_IVTE_Limit_An)
- PSL Error Interrupt Register (PSL_ErrIVTE)
- PSL Control Register (PSL_Control)
- PSL Data Storage Interrupt Status Register (PSL_DSISR_An)
- PSL Data Address Register (PSL_DAR_An)
- PSL Translation Fault Control Register (PSL_TFC_An)
- PSL Error Status Register (PSL_ErrStat_An)
- AFU Control Register (AFU_Cntl_An)
- AFU Error Register (AFU_ERR_An)
• PCIe Type 0 Configuration Space
• CAIA Vendor-Specific PCIe Extended Capability Structure
• AFU Descriptor for compliant AFUs

### 3.3. CAIA Hardware Optional Facilities Tests

The CAIA Test Suite optional facilities tests are generated by writing a functional application that tests the existence of the implemented optional CAIA facilities and verifies the correct behavior. The Open Source CAPI Kernel Driver or equivalent can be used to get access to privileged CAIA facilities, and the CAPI Accelerator Library (libcxl) or equivalent can be used to get access to user-mode CAIA facilities. The following are listed in the CAIA as optional hardware facilities.

- PSL Logical Partition ID Register (PSL_LPID_An)
- PSL AFU Memory Base Address Register (PSL_AMBAR_An)
- PSL AFU Scratch Pad Offset Register (PSL_SPOffset_An)
- PSL Slice Error Register (PSL_SERR_An)
- Hypervisor Accelerator Utilization Record Pointer Register (HAURP_An)
- PSL Context Swap Time Slice Register (PSL_CtxTime_An)
- PSL IVTE Offset Register (PSL_IVTE_Offset_An)
- PSL Context Swap Time Register (PSL_CtxTime)
- AFU Download Control Register (AFU_DLCNTL)
- AFU Download Address Register (AFU_DLADDR)
- PSL Process and Thread Identification Register (PSL_PID_TID_An)
- Context Save/Restore Pointer Register (CSR_P An)
- PSL Process Element Handle Register (PSL_PEHandle_An) (Optional if only dedicated mode is supported)
- PSL WED Register (PSL_WED_An)

### 3.4. Successful Execution of CAIA Required Tests

Each test in the suite of tests should indicate a Return Code 0 for successful completion with no errors. The test harness should summarize the results. It should print error messages indicating what failed if there is a non-zero return code. If there is a non-zero return code the design should be updated to include the missing required facility and the test can be run again.

### 3.5. Successful Execution of CAIA Optional Facilities Tests

Each test in the suite of tests should indicate a Return Code 0 for successful completion with no errors. The test harness should summarize the results. It should print error messages indicating what failed if there is a non-zero return code. If there is a non-zero return code the output should be reviewed to ensure the missing facilities are optional for the Programming mode being supported.
Appendix A. OpenPOWER Foundation overview

The OpenPOWER Foundation was founded in 2013 as an open technical membership organization that will enable data centers to rethink their approach to technology. Member companies are enabled to customize POWER CPU processors and system platforms for optimization and innovation for their business needs. These innovations include custom systems for large or warehouse scale data centers, workload acceleration through GPU, FPGA or advanced I/O, platform optimization for SW appliances, or advanced hardware technology exploitation. OpenPOWER members are actively pursing all of these innovations and more and welcome all parties to join in moving the state of the art of OpenPOWER systems design forward.

To learn more about the OpenPOWER Foundation, visit the organization website at openpowerfoundation.org.

A.1. Foundation documentation

Key foundation documents include:

- Bylaws of OpenPOWER Foundation
- OpenPOWER Foundation Intellectual Property Rights (IPR) Policy
- OpenPOWER Foundation Membership Agreement
- OpenPOWER Anti-Trust Guidelines

More information about the foundation governance can be found at openpowerfoundation.org/about-us/governance.

A.2. Technical resources

Development resources fall into the following general categories:

- Foundation work groups
- Remote development environments (VMs)
- Development systems
- Technical specifications
- Software
- Developer tools

The complete list of technical resources are maintained on the foundation Technical Resources web page.
A.3. Contact the foundation

To learn more about the OpenPOWER Foundation, please use the following contact points:

- General information -- <info@openpowerfoundation.org>
- Membership -- <membership@openpowerfoundation.org>
- Technical Work Groups and projects -- <tsc-chair@openpowerfoundation.org>
- Events and other activities -- <admin@openpowerfoundation.org>
- Press/Analysts -- <press@openpowerfoundation.org>

More contact information can be found at openpowerfoundation.org/get-involved/contact-us.