Arm®v8.4 MPAM Architecture Compliance Suite

Revision: r0p0

Validation Methodology

arm

101321_0000_01_en

Arm®v8.4 MPAM Architecture Compliance Suite

Validation Methodology

Release Information

Document History

Issue	Date	Confidentiality	Change
PJDOC-2042731200-3412	11 May 2018	Non-Confidential	Alpha release
101321-01	13 July 2018	Non-Confidential	Beta release. Note: The document now follows a new numbering format.

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by Arm and the party that Arm delivered this document to.

Unrestricted Access is an Arm internal classification.

Product Status

The information in this document is for a Beta product, that is a product under development.

Web Address

http://www.arm.com

Contents

Arm®v8.4 MPAM Architecture Compliance Suite Validation Methodology

Preface

This preface introduces the *Arm®v8.4 MPAM Architecture Compliance Suite Validation Methodology*.

It contains the following:

About this book

This book is for Arm® MPAM Architecture Compliance Suite.

Product revision status

The r*m*p*n* identifier indicates the revision status of the product described in this book, for example, r*1*p*2*, where:

r*m* Identifies the major revision of the product, for example, r1.

p*n* Identifies the minor revision or modification status of the product, for example, p2.

Intended audience

This user guide is written for engineers who are designing or verifying an implementation of the Arm®v8.4 MPAM Extension Architecture.

Using this book

This book is organized into the following chapters:

Chapter 1 Introduction

Read this chapter for an introduction to the Arm MPAM Architecture Compliance Suite.

Chapter 2 Abstraction layers

Read this chapter for details on the abstraction layers of the MPAM Architecture Compliance Suite.

Chapter 3 Compliance suite build flow

Read this chapter for an overview of MPAM Architecture Compliance Suite environment and build procedure.

Chapter 4 Execution model and flow control

Read this chapter for details on the execution control mode and flow control scheme used by the compliance suite.

Chapter 5 Test validation methodology for standard partitioning control interfaces

Read this chapter for information on the standard partitioning controls available for MPAM Architecture Compliance Suites.

Appendix A Revisions

This appendix describes the technical changes between released issues of this book.

Glossary

The Arm Glossary is a list of terms used in Arm documentation, together with definitions for those terms. The Arm Glossary does not contain terms that are industry standard unless the Arm meaning differs from the generally accepted meaning.

See the *Arm® Glossary* for more information.

Typographic conventions

italic

Introduces special terminology, denotes cross-references, and citations.

bold

Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.

`monospace`

Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.

monospace
: Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.

monospace italic
: Denotes arguments to monospace text where the argument is to be replaced by a specific value.

monospace bold
: Denotes language keywords when used outside example code.

<and>
: Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example:

```
MRC p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2>
```

SMALL CAPITALS
: Used in body text for a few terms that have specific technical meanings, that are defined in the *Arm® Glossary*. For example, IMPLEMENTATION DEFINED, IMPLEMENTATION SPECIFIC, UNKNOWN, and UNPREDICTABLE.

Timing diagrams

The following figure explains the components used in timing diagrams. Variations, when they occur, have clear labels. You must not assume any timing information that is not explicit in the diagrams.

Shaded bus and signal areas are undefined, so the bus or signal can assume any value within the shaded area at that time. The actual level is unimportant and does not affect normal operation.

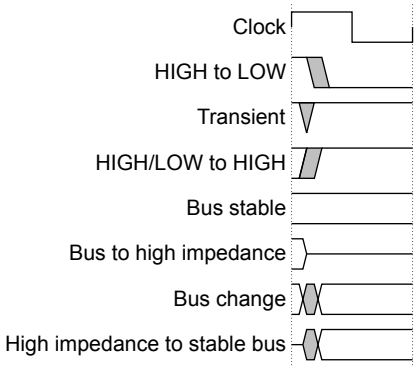

Figure 1 Key to timing diagram conventions

Signals

The signal conventions are:

Signal level
: The level of an asserted signal depends on whether the signal is active-HIGH or active-LOW. Asserted means:
- HIGH for active-HIGH signals.
- LOW for active-LOW signals.

Lowercase n
: At the start or end of a signal name denotes an active-LOW signal.

Additional reading

This book contains information that is specific to this product. See the following documents for other relevant information.

Arm publications

- *Arm®v8.4 MPAM Extension EAC8.0* (ARM-ECM-0485919).

Other publications

- *Advanced Configuration and Power Interface Specification* (ACPI) Version 6.2
- *Unified Extensible Firmware Interface Specification* (UEFI) Version 2.6

Feedback

Feedback on this product

If you have any comments or suggestions about this product, contact your supplier and give:

- The product name.
- The product revision or version.
- An explanation with as much information as you can provide. Include symptoms and diagnostic procedures if appropriate.

Feedback on content

If you have comments on content then send an e-mail to *errata@arm.com*. Give:

- The title *Armv8.4 MPAM Architecture Compliance Suite Validation Methodology*.
- The number 101321_0000_01_en.
- If applicable, the page number(s) to which your comments refer.
- A concise explanation of your comments.

Arm also welcomes general suggestions for additions and improvements.

——— **Note** ———

Arm tests the PDF only in Adobe Acrobat and Acrobat Reader, and cannot guarantee the quality of the represented document when used with any other PDF reader.

———————————

Chapter 1
Introduction

Read this chapter for an introduction to the Arm MPAM Architecture Compliance Suite.

It contains the following section:

1.1 Overview

MPAM *Architecture Compliance Suite* (ACS) is a set of test cases that validate the behavior of standard partitioning and monitoring interfaces recommended by the Arm®v8.4 MPAM Extension Architecture v1.0 specification.

The compliance suite only validates propagation and consumption behaviors for the caches and memory controllers if the PE is already MPAM Extension Architecture complaint.

This Validation Methodology document for MPAM ACS describes the following:

- Compliance test abstraction layers
- Compliance suite build procedure
- Compliance suite execution flow
- Test procedure for standard partioning control interfaces
- Test procedure for standard monitoring control interfaces

The MPAM extensions are optional. So, the compliance suite reads firmware data to discover the system components that implement the MPAM extensions.

Chapter 2
Abstraction layers

Read this chapter for details on the abstraction layers of the MPAM Architecture Compliance Suite.

It contains the following section:

2.1 Abstraction layers

MPAM ACS is divided into three abstraction layers:

- Test Pool
- *Validation Abstraction Layer* (VAL)
- *Platform Abstraction Layer* (PAL)

The following figure shows the interaction of MPAM ACS with UEFI and platform under test.

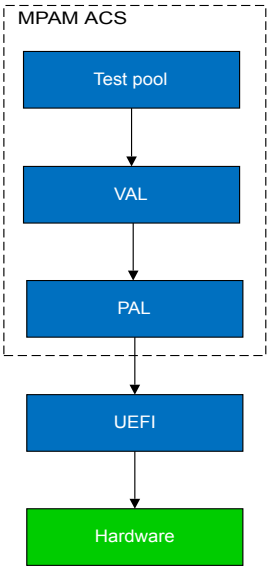

Figure 2-1 Layered software stack for compliance suite

The following table describes the abstraction layers.

Table 2-1 Abstraction layers and their description

Layer	Description
Test pool	Contains the set of directed C-based tests that validate the compliance of the target system against the MPAM Architecture Extension specification. The test uses the APIs provided by the VAL. These tests can be built as a UEFI shell application.
VAL	Provides a set of standard APIs used by the tests. This layer is abstracted above the platform specific details. VAL is generic enough to be used by different partners.
PAL	Provides a standard set of platform hardware requirements to be implemented for running the compliance suite on the partner target platform. This can be built as a PAL library and linked into the final application. These APIs are exposed to the VAL layer and implemented by the partner.

Figure 2-2 illustrates the Compliance Suite interplay with the underlying MPAM master *Memory System Components* (MSCs) and the terminating slave MSCs.

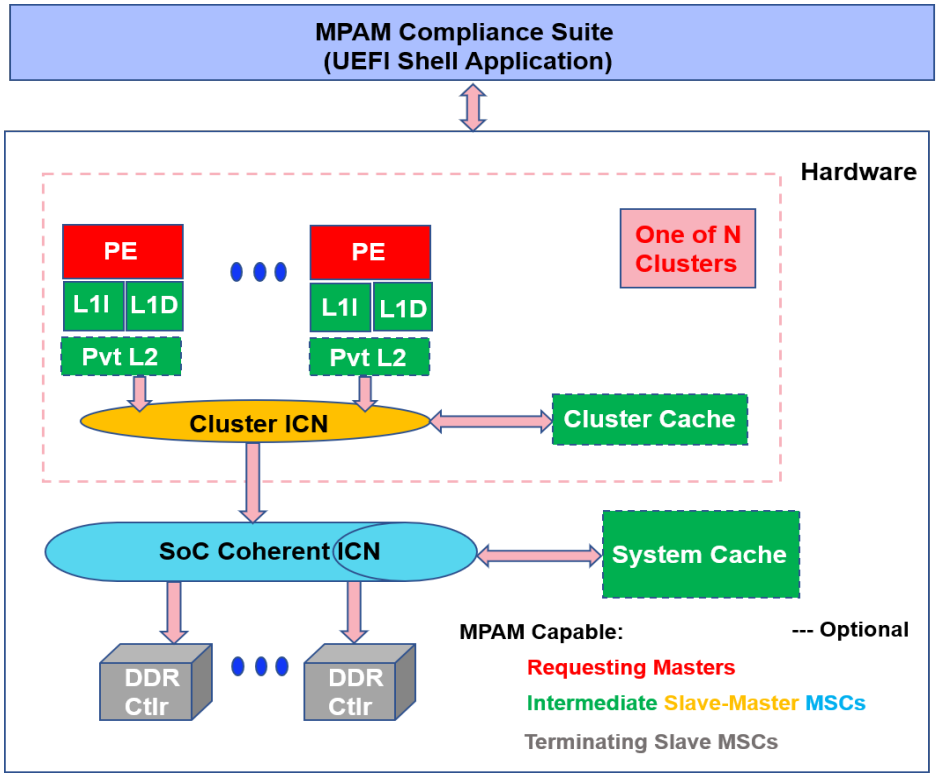

Figure 2-2 Compliance suite interaction with MPAM supported hardware

Chapter 3
Compliance suite build flow

Read this chapter for an overview of MPAM Architecture Compliance Suite environment and build procedure.

It contains the following sections:

3.1 MPAM repository structure

The source code directory structure for MPAM ACS is shown in the following figure.

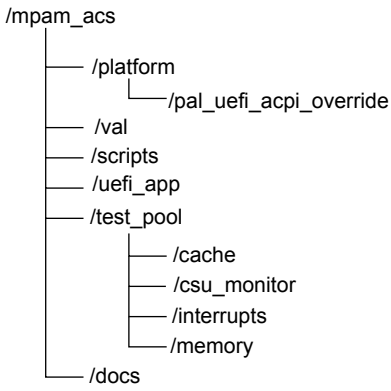

Figure 3-1 MPAM ACS directory structure

The following table describes all the directories.

Table 3-1 MPAM ACS directory structure description

Directory name	Description
`platform`	Platform code targeting UEFI implementation.
`val`	Common code that is used by the tests. Makes calls to PAL as needed.
`scripts`	Scripts written for this suite.
`uefi_app`	UEFI application source to call into the tests entry point.
`test_pool`	Test case source files for test suite.
`docs`	Documentation.

3.2 Test build

To build MPAM compliance suite as a UEFI Shell application, a UEFI EDK2 source tree is required.

For more information, see *https://github.com/ARM-software/arm-enterprise-acs/blob/master/mpam/README.md*.

The build steps for the compliance suite to be compiled as a UEFI shell application are at: *https://github.com/ARM-software/arm-enterprise-acs/blob/master/mpam/README.md*

3.3 MPAM ACS build procedure

To build MPAM ACS suite as an EDK2 shell application, follow these steps:

Procedure

1. Download the compressed cross-compiler tar ball.

```
wget https://releases.linaro.org/components/toolchain/binaries/7.1-2017.05/aarch64-linux-gnu/gcc-linaro-7.1.1-2017.05-x86_64_aarch64-linux-gnu.tar.xz
```

2. Unzip the downloaded tar ball.

```
xz -d gcc-linaro-7.1.1-2017.05-x86_64_aarch64-linux-gnu.tar.xz
```

3. Extract the cross-compiler binaries from the unzipped tar ball.

```
tar -xf gcc-linaro-7.1.1-2017.05-x86_64_aarch64-linux-gnu.tar
```

4. Export the cross-compiler. You must use the absolute path while doing this.

```
export GCC49_AARCH64_PREFIX=/path/to/CROSS_COMPILER/gcc-linaro-7.1.1-2017.05-x86_64_aarch64-linux-gnu/bin/aarch64-linux-gnu-
```

5. Checkout the EDK2 tree. Install the prerequisite packages to build the EDK2 tree.

```
git clone https://github.com/tianocore/edk2.git
```

6. Setup EDK2 environment. This must be setup each time for a new terminal.

```
cd /path/to/edk2
. edksetup.sh
```

7. Build EDK2 tools. Note that the `make -C BaseTools/Source/C` command must be executed once after syncing the EDK2 tree.

```
cd /path/to/edk2/
make -C BaseTools/Source/C
```

8. Checkout MPAM ACS suite.

```
git clone https://github.com/ARM-software/arm-enterprise-acs.git
```

9. Create a symbolic link for `mpam` in the EDK2 tree. You must use absolute paths while creating the symbolic links.

```
cd /path/to/edk2/
ln -s /path/to/arm-enterprise-acs/mpam AppPkg/Applications/mpam
```

10. Add following two libraries to `[LibraryClasses.common]` section in `path/to/edk2/ShellPkg/ShellPkg.dsc`

```
MpamValLib|AppPkg/Applications/mpam/val/MpamValLib.inf
MpamPalLib|AppPkg/Applications/mpam/platform/pal_uefi_acpi_override/MpamPalLib.inf
```

11. Add `AppPkg/Applications/mpam/uefi_app/MpamAcs.inf` in `[Components]` section in `path/to/edk2/ShellPkg/ShellPkg.dsc`
12. Compile MPAM ACS test suite.

```
cd /path/to/edk2/
source AppPkg/Applications/mpam/scripts/acsbuild.sh
```

Chapter 4
Execution model and flow control

Read this chapter for details on the execution control mode and flow control scheme used by the compliance suite.

It contains the following section:

4.1 Compliance suite execution flow

The following figure shows the execution model of compliance suite and the flow control used.

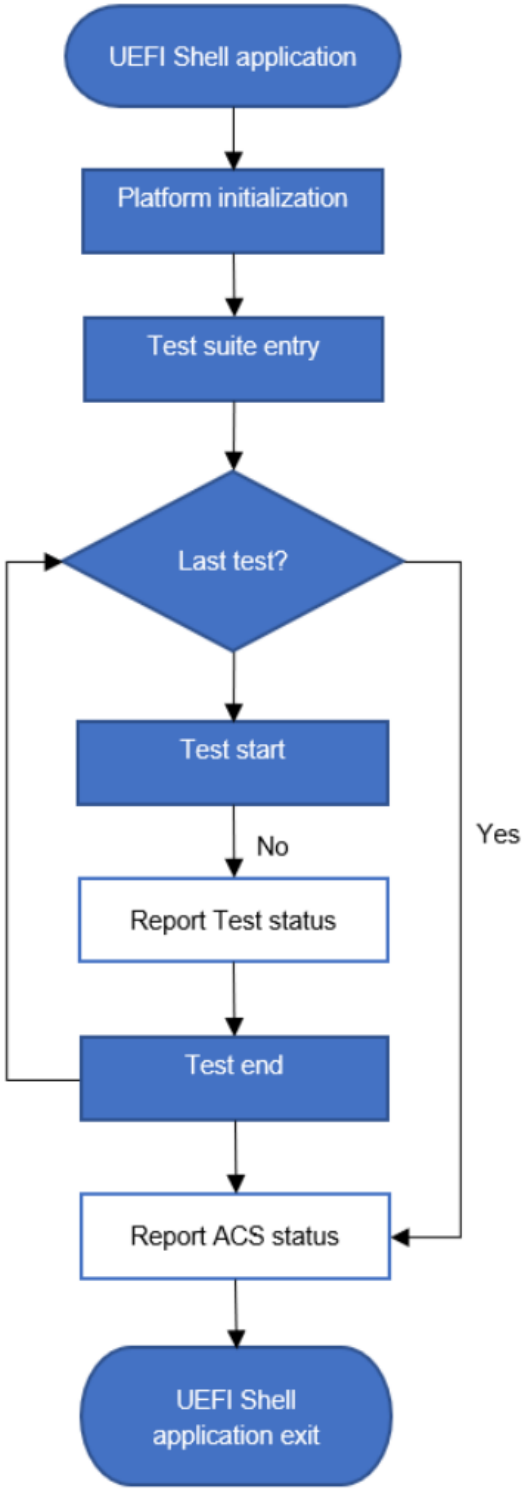

Figure 4-1 Compliance suite execution flow

Chapter 5
Test validation methodology for standard partitioning control interfaces

Read this chapter for information on the standard partitioning controls available for MPAM Architecture Compliance Suites.

It contains the following sections:

5.1 Standard partitioning control interfaces

The standard partitioning controls supported in MPAM compliance are listed below. All the tests run on a single PE.

- Cache portion partitioning
- Cache maximum capacity partitioning
- Memory bandwidth portion partitioning
- Minimum bandwidth limit partitioning
- Maximum bandwidth limit partitioning

5.2 Cache portion partitioning

Consider two scenarios allocated with extreme system cache portions and measure DDR latencies for a mem copy of fixed chunk size.

Figure 5-1 shows the procedure for a system in which L4 is the highest-level cache with 8-way set associativity and implements way-based portion partitioning.

Scenario 1

Configure the Cache Portion Bit Map partition config register `MPAMF_CPBM` such that three-fourth of the maximum sized MPAM cache is portion partitioned and allocated to this scenario. Perform a DDR mem copy of size equal to three-fourth of system cache size. This ensures that all these memory locations are available in different levels of caches. Repeat the mem copy of the same chunk and measure the latency.

Scenario 2

Configure the Cache Portion Bit Map partition config MPAMF_CPBM register such that one-fourth of system cache is portion partitioned and allocated to this scenario.

Repeat the procedure in scenario 1 and measure the DDR latency. Cache misses and evictions are expected in this case since one-fourth system cache cannot accommodate the input traffic that is three times greater.

Note

- PMU cycle count register `PMCCNTR` can be used to measure the mem copy latencies since they are very small.
- Latency measurements are done only during mem copy repeat and not during initial copy.
- Use *Processor Properties Topology Table* (PPTT) ACPI table to know the cache topology for the PE.
- Configure memory partitioning controls to get 100% bandwidth during each test scenario.
- The number of cache misses as read from a PMU counter for each cache can be used to show the increase in number of misses. Future releases of MPAM ACS can explore this option.

Most system caches have coarse partitioning because they may implement portion partitioning by cache ways. The last-level caches could have 8 to 32 ways. If there are 32 ways to work with, each way is about 3.3% of the cache; one-fourth partition gets 25% of cache and 8-way associativity. This is good associativity. In an 8-way system cache, one-fourth partition gets just 2-way associativity. This is low associativity. So, as the number of ways goes down, way partitioning impacts the cache performance drastically.

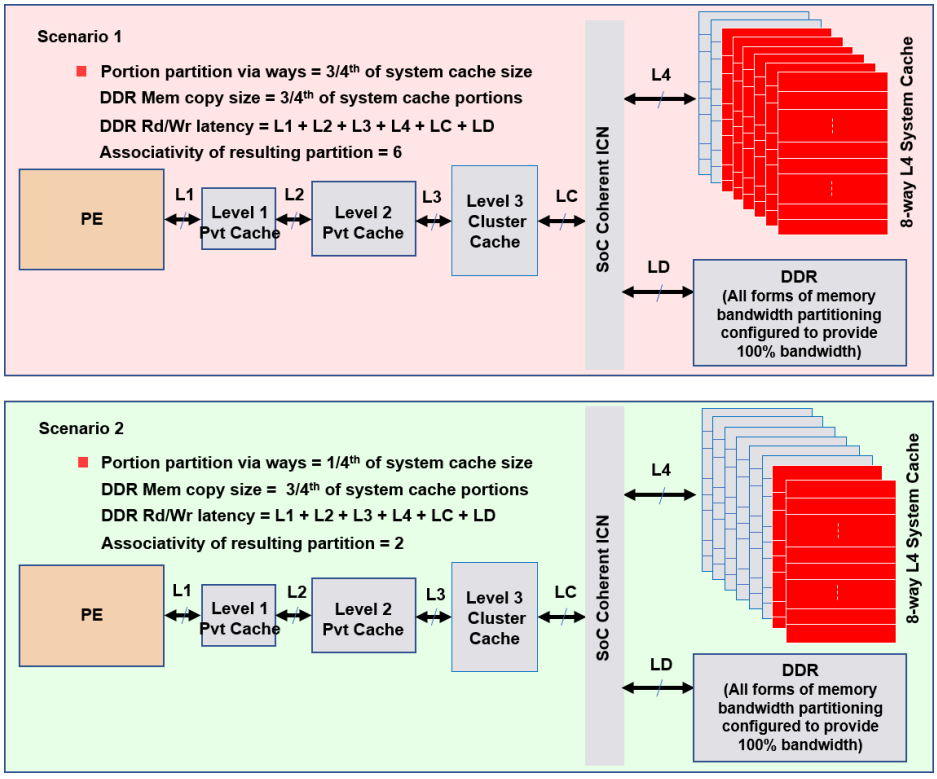

Figure 5-1 Test procedure to validate cache portion partitioning

5.3 Cache maximum capacity partitioning

In this scenario, the partitions are created by dividing the cache capacity.

However, in cache portion partitioning, the partitions are created from the portions. So, cache portion partitioning test procedure can be reused in this scenario. Use `MPAMF_CMAX` cache maximum capacity partition configuration register to configure the capacity partitions for each scenario that is described in *5.1 Standard partitioning control interfaces* on page 5-23. Similar latency comparisons among scenarios can be made here to declare this test as pass or fail.

Note

When cache portion partitioning is implemented through cache ways with small cache resource (less number of ways) allocated to a partition, portion partitioning reduces the cache's associativity of the partition to only one or two ways. This affects performance. With cache maximum capacity partitioning, even a small allocation can get high associativity in sets where needed. The result should be considerably higher performance.

5.4 Memory bandwidth portion partitioning

Memory bandwidth portion partitioning uses the same principle as cache portion partitioning.

This test scenario is independent of DDR bandwidth and number of bits w available for bandwidth portioning, both of which are IMPLEMENTATION DEFINED. `MPAMCFG_MBW_PBM` config register is used to create two bandwidth portions of sizes 1/2^w and 1/2^(w-1), where w can range from 1 to 12. The input traffic is fixed to one Megabyte so that the latency for any value of w portion bits can be measured using a PMU counter.

The latency in case of 1/2^w bandwidth portion is double the latency for 1/2^(w-1) bandwidth portion.

Note

- For memory bandwidth-related test cases, configure cache partitioning controls to get 100% cache availability during each test scenario.
- Prefill the caches with some data so that all the input traffic is read from DDR before making bandwidth latency measurements.
- Use PPTT ACPI table to know the DDR memory node topology.

5.5 Maximum bandwidth partitioning

For this test case, set `HARDLIM =1` in `MPAMCFG_MBW_MAX` config register.

If `HARDLIM` is set, the partition is prevented from using any bandwidth above the maximum bandwidth limit.

Create two scenarios in which the first one takes an input traffic of size equivalent to the partition's maximum bandwidth limit. In the second scenario, the input traffic is double the size of the partition's maximum bandwidth limit. Set the maximum bandwidth limit to 1/2^w and measure the latency in both the scenarios. The second scenario has twice the latency of the first scenario.

Note

Based on the number of fractional bits for maximum bandwidth, limiting can be computed. For example, the partition's maximum bandwidth limit corresponding to 1/2^w is 2 GBps for w=8 and DDR bandwidth of 512 GBps. It is 32 GBps for w=4 and DDR bandwidth of 512 GBps. As generating such high traffic is not viable through the PEs, the compliance suite assumes only one DDR channel to be configured and the PEs together can generate traffic to contend the memory channel.

Appendix A
Revisions

This appendix describes the technical changes between released issues of this book.

It contains the following section:

- *A.1 Revisions* on page Appx-A-30.

A.1 Revisions

Table A-1 Issue PJDOC-2042731200-3412

Change	Location	Affects
This is the first revision of the document.	-	All revisions

Table A-2 Differences between Issue PJDOC-2042731200-3412 and Issue 101321-01

Change	Location	Affects
		All revisions