

Arm® Authenticated Debug Access Control Test Suite

Version 0.5

User Guide



Arm® Authenticated Debug Access Control Test Suite

User Guide

Copyright © 2021 Arm Limited or its affiliates. All rights reserved.

Release Information

Document History

| Issue | Date | Confidentiality | Change |
|---------|--------------|------------------|---------------|
| 0005-01 | 30 June 2021 | Non-Confidential | Alpha release |

Non-Confidential Proprietary Notice

This document is protected by copyright and other related rights and the practice or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of Arm. **No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.**

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether implementations infringe any third party patents.

THIS DOCUMENT IS PROVIDED “AS IS”. ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, third party patents, copyrights, trade secrets, or other rights.

This document may include technical inaccuracies or typographical errors.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

This document consists solely of commercial items. You shall be responsible for ensuring that any use, duplication or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word “partner” in reference to Arm’s customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

If any of the provisions contained in these terms conflict with any of the provisions of any click through or signed written agreement covering this document with Arm, then the click through or signed written agreement prevails over and supersedes the conflicting provisions of these terms. This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of the Agreement shall prevail.

The Arm corporate logo and words marked with ® or ™ are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners. Please follow Arm’s trademark usage guidelines at <https://www.arm.com/company/policies/trademarks>.

Copyright © 2021 Arm Limited (or its affiliates). All rights reserved.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

(LES-PRE-20349)

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 an Alpha product, that is a product under development.

Web Address

developer.arm.com

Inclusive language commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used language that can be offensive. Arm strives to lead the industry and create change.

This document includes language that can be offensive. We will replace this language in a future issue of this document.

To report offensive language in this document, email terms@arm.com.

Contents

Arm® Authenticated Debug Access Control Test Suite User Guide

Preface

| | |
|------------------------------|---|
| <i>About this book</i> | 6 |
|------------------------------|---|

Chapter 1

Introduction

| | | |
|-----|-----------------------------------|------|
| 1.1 | <i>Abbreviations</i> | 1-9 |
| 1.2 | <i>Introduction to ADAC</i> | 1-10 |
| 1.3 | <i>ADAC test suite</i> | 1-11 |

Chapter 2

Validation methodology

| | | |
|-----|---|------|
| 2.1 | <i>Scope of the ADAC test suite</i> | 2-13 |
| 2.2 | <i>Tool requirements</i> | 2-14 |
| 2.3 | <i>Prerequisites</i> | 2-15 |
| 2.4 | <i>Test methodology</i> | 2-16 |

Appendix A

Revisions

| | | |
|-----|------------------------|-----------|
| A.1 | <i>Revisions</i> | Appx-A-20 |
|-----|------------------------|-----------|

Preface

This preface introduces the *Arm® Authenticated Debug Access Control Test Suite User Guide*.

It contains the following:

- [About this book on page 6.](#)

About this book

This book describes the user guide for Arm® Authenticated Debug Access Control test suite.

Using this book

This book is organized into the following chapters:

Chapter 1 Introduction

This chapter provides information on ADAC test suite.

Chapter 2 Validation methodology

This chapter describes the validation methodology used for the ADAC test suite.

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.

Additional reading

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

Arm publications

- *PSA Cryptography API* (IHI 0086)
- *Authenticated Debug Access Control Specification* (DEN0101)

Other publications

None.

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 support-psa-arch-tests@arm.com. Give:

- The title *Arm Authenticated Debug Access Control Test Suite User Guide*.
- The number 102545_0005_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.

Other information

- *Arm® Developer*.
- *Arm® Documentation*.
- *Technical Support*.
- *Arm® Glossary*.

Chapter 1

Introduction

This chapter provides information on ADAC test suite.

It contains the following sections:

- [1.1 Abbreviations](#) on page 1-9.
- [1.2 Introduction to ADAC](#) on page 1-10.
- [1.3 ADAC test suite](#) on page 1-11.

1.1 Abbreviations

This section lists the abbreviations used in this document.

Table 1-1 Abbreviations and expansions

| Abbreviation | Expansion |
|--------------|------------------------------------|
| ADAC | Authenticated Debug Access Control |
| PAL | Platform Abstraction Layer |
| PSA | Platform Security Architecture |
| RDDI | Remote Device Debug Interface |
| RoT | Root of Trust |
| SPM | Secure Partition Manager |
| SUT | System Under Test |
| VAL | Validation Abstraction Layer |

1.2 Introduction to ADAC

Authenticated Debug Access Control (ADAC) is a protocol that provides a way to use strong authentication to restrict device debug access to authorized entities only.

ADAC defines communication between two entities:

1. The device being debugged, which is the target.
2. The debugger itself, which is the host.

The ADAC specification addresses functional layers that sit above the physical debug link. Physical access to the target device is required for debug link operation.

The default mechanism for ADAC authentication relies on a challenge-response protocol where the target challenges the host and verifies the response validity. The response to the challenge is a signed authentication token, also called a debug token. The key used by the host to generate the signature must be trusted by the target, possibly through a chain of trusted entities already provisioned on the target. The ADAC specification defines a certificate format to build trust chains and offer the flexibility to deal with complex scenarios.

For more information on ADAC, see the [Authenticated Debug Access Control Specification](#).

1.3 ADAC test suite

The ADAC test suite checks if a device-side implementation conforms to the behavior described in the ADAC specification. For host-side conformance to the ADAC specification, Arm is working directly with partner tool vendors to guarantee compatibility.

The ADAC tests are self-checking and portable C-based tests with directed stimulus. These tests are expected to run on the host platform only. The test suite does not require any debugger but uses Arm Remote Device Debug Interface (RDDI) libraries to communicate with the target being tested. Arm RDDI libraries can be obtained as part of the [Arm Development Studio](#).

The tests drive the ADAC commands from the host platform and verify the response obtained from the target. The tests are open source and available with an Apache v2.0 license, allowing for external contribution.

Chapter 2

Validation methodology

This chapter describes the validation methodology used for the ADAC test suite.

It contains the following sections:

- *2.1 Scope of the ADAC test suite* on page 2-13.
- *2.2 Tool requirements* on page 2-14.
- *2.3 Prerequisites* on page 2-15.
- *2.4 Test methodology* on page 2-16.

2.1 Scope of the ADAC test suite

The scope of the ADAC test suite is as follows:

1. Validate that the tested target implements all ADAC commands.
2. Check that the response received for a given ADAC command is one of the defined expected responses.
3. Test commands, errors, and success criteria mentioned in the ADAC specification.

2.2 Tool requirements

The ADAC test suite is developed in C and compiled for the host execution environment.

The tests communicate with the target using the libraries defined by the RDDI interface. The RDDI libraries are available in your Arm Development Studio install directory (<Install Directory>/sw/debugger/RDDI).

Software prerequisites on the host platform:

- CMake v3.1 or greater
- GNU Arm Embedded Toolchain version 9-2019-q4-major
- Python 3.7 or later, with pip and virtualenv.
- RDDI header files and redistributable libraries.
- Access to PSA-ADAC source code repository, as an alpha-specific temporary requirement.

Note

RDDI layer integration is not included in this release.

2.3 Prerequisites

The ADAC tests require the following setup for the device to execute the tests on a host platform.

- Credentials must be provisioned in the device and the public key of a root certificate must either be present on the target in hashed form or in its entirety and accessible to the ADAC target code.
- A set of tests in a given run only verifies a single cryptosystem algorithm. To test multiple cryptosystems, target devices must be provisioned with adequate credentials and the test suite should re-run for each cryptosystem to be tested.

2.4 Test methodology

This chapter describes the test methodology for the ADAC test suite.

This section contains the following subsections:

- [2.4.1 Test layering details on page 2-16.](#)
- [2.4.2 Directory structure on page 2-16.](#)
- [2.4.3 Build and execution flow on page 2-17.](#)
- [2.4.4 Test status reporting on page 2-18.](#)
- [2.4.5 Additional notes on page 2-18.](#)

2.4.1 Test layering details

The ADAC tests use a layered software stack approach to enable porting across different test platforms.

The constituents of the layered stack are:

1. Test suite
2. Platform Abstraction Layer (PAL)
3. Validation Abstraction Layer (VAL)

Table 2-1 Test layering details

| Layer | Description |
|------------|--|
| Test suite | This layer is a collection of targeted tests. It verifies if the device behavior conforms to the ADAC specification. These tests map to one or more scenarios identified in the scenario document. They use interfaces that are provided by the VAL. |
| PAL | This layer is the closest to hardware and is aware of underlying hardware details. It must be ported or tailored to specific host platform. It implements the debug link layer responsible for communication with the target. |
| VAL | This layer contains subdirectories for the VAL libraries. It provides a uniform and consistent view of the available test infrastructure to the tests in the test pool. This layer is not ported when the underlying hardware changes. |

Note

This release provides a reference implementation of the PAL layer using Unix sockets to communicate with the target. For more information on implementing the link layer, see the [Porting Guide](#).

2.4.2 Directory structure

The following figure shows the top-level directory structure of Secure debug within the `psa-arch-tests` Git repository.

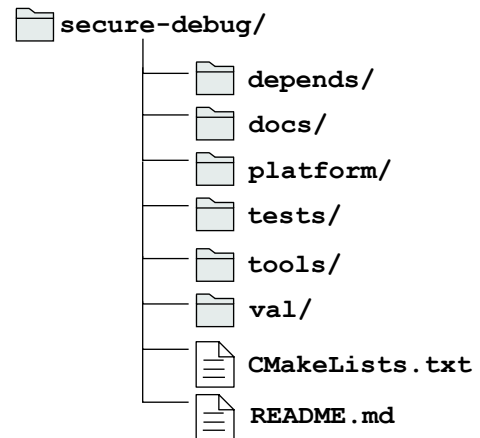


Figure 2-1 Secure debug directory structure

| | |
|-----------------------|---|
| depends | contains the pre-built shared libraries that implement standard link layer communication protocols. You can add your custom debug interface libraries and reference it for your platform layer API. For more information on the platform layer, see the Porting Guide . |
| docs | contains the User Guide (this document), Porting Guide and scenarios to be tested. |
| platform | contains files to form the PAL. It must be ported or tailored to specific host platform. |
| tests | has the ADAC sub-suite containing tests that verify if the device behavior conforms with the ADAC specification. |
| tools | contains makefiles and scripts that are used to generate test binaries. |
| val | contains subdirectories and files for building the VAL libraries. This layer provides a uniform and consistent view of the available test infrastructure to the tests. |
| CMakeLists.txt | contains information about CMake build support. |
| README.md | README file for PSA ADAC test suite. |

2.4.3 Build and execution flow

This section provides steps to build and execute the ADAC test suite for a given host platform.

Build steps

The steps and the dependencies for building the host-side executable are described in detail in the README file. The following steps compile and generate the test image for the ADAC suite:

```
cd psa-arch-tests/secure-debug
mkdir <host_build_dir>
cd <host_build_dir>
cmake ../ -G"Unix Makefiles" - DTARGET=<target_name> -DSUITE=<suite_name>
```

For example,

```
cmake ../ -G"Unix Makefiles" -DTARGET=native -DSUITE=ADAC
cmake --build .
```

This generates the `psa_adac_test` executable in the host-build directory.

Execution steps

The executable should be provided with the host-side credentials (key and certificate chain) as command-line arguments. Depending on the debug link layer used, additional command-line parameters should be provided.

```
./psa_adac_test <path_to_key_file> <path_to_certificate> / <optional list to link-layer details>
```

Note

The README file describes the details for executing the test on a native Unix-host platform.

2.4.4 Test status reporting

When the test suite is run on a host platform, each successfully run test of the suite must report either PASS or SKIP.

The following is an example code of a successful test pass.

```
**** PSA Architecture Test Suite - Version 1.2 ****

Running .. Secure Debug Suite
*****
TEST: 801 | DESCRIPTION: Testing ADAC Protocol Host API | UT: psa_challenge
Sending challenge request
Receiving challenge
status = 0x0000, data_count = 9
val_parse_response : 233 : debug : host :challenge 0000: 2c486475b5a6661035c68dea731d0214
val_parse_response : 233 : debug : host :challenge 0010: 54949be72f27d31fcea2b138da251f9f
Sending challenge request
Receiving challenge
status = 0x0000, data_count = 9
val_parse_response : 233 : debug : host :challenge 0000: e5bc013aa4a152eadc001dafd584eb04
val_parse_response : 233 : debug : host :challenge 0010: f701c0cc21f80e61aced58e7179cd187
Challenge response obtained is unique

TEST RESULT: PASSED
```

The following code displays the status of all the tests scheduled in a single execution run.

```
*****
***** Secure Debug Suite Report *****
TOTAL TESTS      : 2
TOTAL PASSED     : 2
TOTAL SIM ERROR  : 0
TOTAL FAILED     : 0
TOTAL SKIPPED    : 0
*****
Entering standby..
```

2.4.5 Additional notes

The following are a few additional points to consider:

- The data structures and the packet format described in the specification are defined in the `psa_adac.h` header file within the `psa-adac` repository.
- The `val_adac.h` defines the API for the following tasks:
 - Managing the host credentials (uses mbedtls services).
 - Building and constructing ADAC commands.
 - Parsing the response obtained from the debug target.
- The PAL API which is responsible for implementing the debug link layer is defined in `pal_interfaces.h`. These APIs must be ported for your host platform. For more information on API porting, see the [Porting Guide](#).
- The `psa-adac` repository provides a tool for generating keys and certificates. The same is used to generate credentials for authenticating the host.

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-20.](#)

A.1 Revisions

This section consists of all the technical changes between different versions of this document.

Table A-1 Issue 0100-01

| Change | Location |
|---|----------|
| This is the first revision of the document. | - |