arm

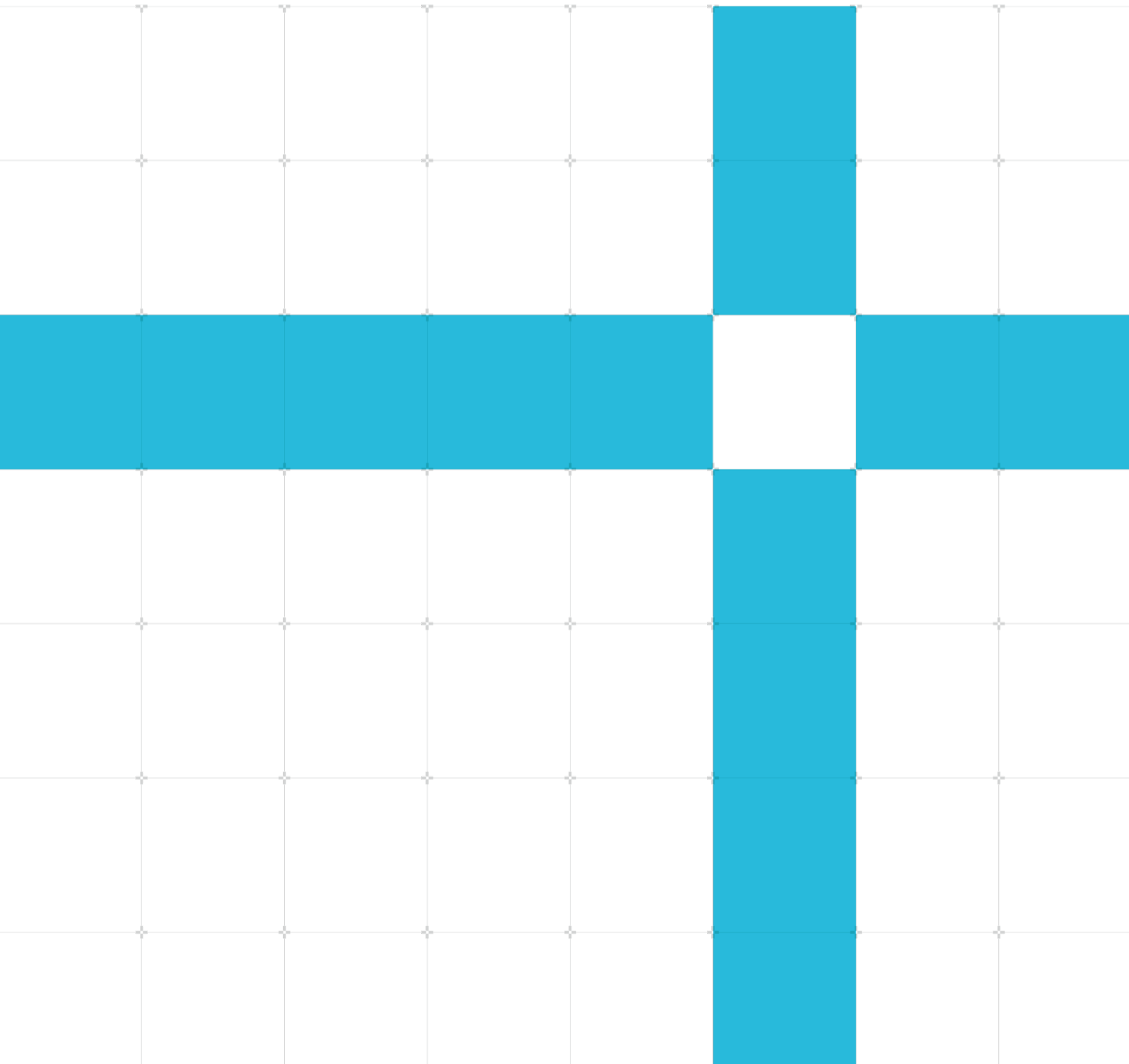
Arm Compiler for Linux

Version: 23.04.1

Release Note

Non-Confidential

Issue 00
107578_2304.1_00



Non-Confidential Proprietary Notice

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 Final, that is for a developed product.

Web Address

https://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.

We believe that this document contains no offensive terms. To report offensive language in this document, email terms@arm.com.

Contents

1 Release overview

1.1 Product description

The Arm® Compiler for Linux 23.04.1 suite provides a complete compiling environment for natively developing and tuning your server and HPC applications on Arm-based platforms.

The suite contains the following packages:

- Arm® C/C++/Fortran Compiler 23.04.1

  Arm Compiler is a Linux user-space C/C++ and Fortran compiler tailored for scientific computing, HPC, and enterprise workloads.

- Arm® Performance Libraries 23.04.1

  Arm® Performance Libraries contains optimized math functions, such as linear algebra and Fast Fourier Transforms, for Arm AArch64 implementations, including those with SVE. It is compatible with Arm® C/C++/Fortran Compiler 23.04.1 and GCC 12.2.0.

  Arm® Performance Libraries is optimized for a number of microarchitectures. The latest information is available on the Arm Developer website:

  https://developer.arm.com/Tools%20and%20Software/Arm%20Compiler%20for%20Linux#Supported-Devices

- GCC 12.2.0

  For convenience, and to provide the optimal experience of using Arm Performance Libraries and GCC on the latest Arm server and HPC systems, a build of GCC 12.2.0 is provided. The GCC 12.2.0 build is also provided for OpenMP/libgfortran compatibility with Arm Performance Libraries.

  As a GNU tool suite, the GPL-licensed source code can be downloaded separately.

1.2 Release status

This is the 23.04.1 release of the Arm® Compiler for Linux software.

These deliverables are being released under the terms of the agreement between Arm and each licensee (the “Agreement”). All planned verification and validation is complete. The release is suitable for volume production under the terms of the Agreement.

2 Release contents

The following sub-sections describe:

- The product parts that are delivered as part of this release.
- Any changes since the previous release.
- Any known issues and limitations that exist at the time of this release.

2.1 Deliverables

- Arm® C/C++/Fortran Compiler 23.04.1
- Arm® Performance Libraries 23.04.1
- GCC 12.2.0
- Release Notes (this document)
- Documentation

  Arm® Compiler for Linux reference guides are available in <install_location>/<package_name>/share. The guides that are in the '/share' location are also available on the Arm Developer website:

  https://developer.arm.com/Tools%20and%20Software/Arm%20Compiler%20for%20Linux#Resources

  The same Arm Developer web page also contains links to tutorials, installation guides, and application porting guides.

  Documentation and release notes might change between product releases. For the latest documentation bundle, check the product download page.

  Arm tests PDFs only in Adobe Acrobat and Acrobat Reader. Arm cannot guarantee the quality of its PDFs when used with any other PDF reader. Adobe reader products are available at https://www.adobe.com.

2.2 Differences from previous release

Arm® Compiler for Linux 23.04.1 includes various internal changes that resolve defects and improve performance.

The following subsections describe the significant differences from the previous release of Arm® Compiler for Linux.

Additions and changes:

This section describes the new features or components added, or any significant technical changes to features or components, in the 23.04.1 release.

- Arm® C/C++/Fortran Compiler 23.04.1:
  - The underlying LLVM technology in ACfL 23.04.1 is updated to LLVM 16.0.2.

This section describes the new features or components added, or any significant technical changes to features or components, in the 23.04 release.

- Arm® Compiler for Linux suite 23.04:
  - Upgraded the version of GCC, from GCC 11.2.0 to GCC 12.2.0
  - Arm® Compiler for Linux 23.04 simplifies the path names of the installed directories to use a clearer pattern with less redundancy. The new pattern is as follows: <component>-<version>_<OS>. For paths to performance libraries directories, the name will also contain the compiler which built the libraries, i.e. arm-linux-compiler or gcc. For example, on a RHEL 9 operating system, you will find the following installed folders:

    * arm-linux-compiler-23.04_RHEL-9
    * armpl-23.04.0_RHEL-9_arm-linux-compiler
    * armpl-23.04.0_RHEL-9_gcc
    * gcc-12.2.0_RHEL-9
  - Arm® Compiler for Linux 23.04 adds support for the following new Linux platforms:
    * Red Hat Enterprise Linux 9 (RHEL 9)
    * Ubuntu 22.04 LTS
    * Amazon Linux 2 and Amazon Linux 2023
  - Arm® Compiler for Linux is no longer supported on Ubuntu 18.04 LTS
- Arm® C/C++/Fortran Compiler 23.04:
  - Arm® Compiler for Linux 23.04 has upgraded its base clang version from Clang 13.0.0 to Clang 16.0.0 (of the LLVM Compiler project), which provides improved performance and stability.

    More details about the changes introduced since Arm® Compiler for Linux 22.1 can be found in the release notes for LLVM releases 14-16:
    * https://releases.llvm.org/14.0.0/tools/clang/docs/ReleaseNotes.html
    * https://releases.llvm.org/15.0.0/tools/clang/docs/ReleaseNotes.html
    * https://releases.llvm.org/16.0.0/tools/clang/docs/ReleaseNotes.html
  - Enhanced vectorization support when targeting systems with the Armv8.3-A Floating-point Complex Number extensions (FEAT_FCMA). ACfL can now vectorize loops containing operations on variables defined with complex data types, emitting floating point FCADD and FCMLA instructions.
  - Added support for the Neoverse V2 core. To enable architectural optimizations for Neoverse V2, use '-mcpu=neoverse-v2' on your compiler command line.
  - Support has been added to the compiler for producing a dynamically linked binary but with all ACfL libraries statically linked. This enables creating binaries for redistribution without having to ship libraries from ACfL like libamath.so.

    To enable this feature, add '-static-arm-libs' to your compiler command line when linking.

    Note: the -static-arm-libs compilation flag may not work correctly if an attempt is made to link with system libraries which are not built as position independent executables. In such a situation, add '-nopie' to the link commandline.
  - The default behavior of armclang, armclang++ and armflang with regards to generating position independent code has changed in Arm® Compiler for Linux 23.04. The default setting is now '-fPIE', that is to generate a position independent binary. Note, the GCC toolchain provided continues to default to '-fno-pie'
  - The default value of the '-fsimdmath' compiler option has changed. From 23.04, armclang/armclang++ enables '-fsimdmath' by default when compiling with optimization enabled, enabling the compiler to vectorize loops containing calls to supported math functions. To disable this optimization, add '-fno-simdmath' to your compiler command line.
  - armclang is now more strict in checking for implicit function and variable declarations and will now emit errors like "call to undeclared function". These errors can be downgraded to warnings using '-Wno-error=implicit- function-declaration'. See the LLVM 15 release notes for more detail https://releases.llvm.org/15.0.0/tools/clang/docs/ReleaseNotes.html
  - Arm Compiler 23.04 removes support for the "scalable" extension to the '#pragma omp declare variant' construct, previously in Arm Compiler 22.1.

    This affects code containing constructs like below:


```
#pragma omp declare variant(UserSinCos) \
match(construct = {simd(notinbranch, linear(sin),
linear(cos)) }, device = {isa("sve") }, implementation =
{extension("scalable") })
```

For details on the removed feature, see:

https://developer.arm.com/documentation/101458/2210/Optimize/Vector-routines-support/How-to-declare-custom-vector-routines-in-Arm-C-C—Compiler?lang=en

- Arm® Compiler for Linux 23.04 simplifies the naming scheme for the AArch64 math and strings processing libraries. Architecture specific suffixes have been removed from library names. ACfL no longer links with microarchitecture specific libamath or libastring, and the package does not include these libraries anymore. If you are manually linking to these libraries in your linker command line, the new compiler flags to use should be: '-lamath' or '-lastring'.
- The default C++ standard has changed from '-std=gnu++14' to '-std=gnu++17', that is C++17 and conforming GNU extensions. Projects incompatible with C++17 can add '-std=gnu++14' to their build settings to restore the previous behavior.
- The compiler option -Warm-extensions has been removed as ACfL does not support any extra Arm specific language extensions.

- Arm® Performance Libraries 23.04:
  - New routines for sparse linear algebra, including parallel optimizations:
    - Sparse matrix functionality:
      - Triangular matrix solve: armpl_spsv_exec_*
    - Introduction of a new sparse vector type, armpl_spvect_t. Routines for operations on sparse vectors:
      - Dot product: armpl_spdot*_exec_*
      - AXPBY: armpl_spaxpby_exec_*, armpl_spwaxpby_exec_*
      - Plane rotation: armpl_sprot_exec_*
      - Utilities: armpl_spvec_gather_exec_*, armpl_spvec_scatter_exec_*
    - See examples and online documentation for details.
  - Support for LAPACK version 3.11.0.
  - Increased performance for:
    - Small ?GEMM problems.
    - Large parallel thread counts for all BLAS routines across microarchitectures.
    - FFT functions.
  - The GCC version of the library shipped with Arm® Compiler for Linux is now compatible with GCC 12.2 instead of 11.2.
    - Note that builds of Arm® Performance Libraries compatible with many versions of GCC are available separately online.

Resolved issues:

Describes any technical issues that are resolved in the 23.04.1 release.

- Arm® C/C++/Fortran Compiler 23.04.1:
  - Fixed a bug where armclang would crash with exit code 139 when compiling code using SVE types inside an OpenMP region with -fopenmp and targeting an SVE platform.
- Arm® Performance Libraries 23.04.1:

  - Integer overflow fixed in armpl_spmat_export* functions in lp64 libraries.

Describes any technical issues that are resolved in the 23.04 release.

- Arm® Compiler for Linux suite 23.04:
  - Improved RPM packaging for the GCC toolchain. We included the full list of symbols provided by the GCC toolchain, so that RPM package dependencies can be properly resolved.
- Arm® C/C++/Fortran Compiler 23.04:
  - ACfL has improved optimization of calls to log with arguments of type ‘long double’ in C and C++ code. This improves the performance of WarpX.
- Arm® Performance Libraries 23.04:
  - pkgconfig files renamed and relocated.

2.3 Known limitations

The following subsection describes any issues that are known at the time of this release.

Open technical issues:

There are no open technical issues in the 23.04.1 release.

3 Get started

This section describes information to help you get started with accessing, setting up, and using Arm® Compiler for Linux.

For more information, see the Get Started information on the Arm Developer website:

https://developer.arm.com/documentation/101458/2300/Get-started

3.1 Licensing information

Use of Arm® Compiler for Linux is subject to the terms and conditions of the applicable End User License Agreement ("EULA"). A copy of the EULA can be found in the 'license_terms' folder of your product installation.

You do not require a license to use this Arm® Compiler for Linux package.

3.2 Prerequisites

If any of the following tools are not already installed by your Linux distribution, you must install them before installing Arm® Compiler for Linux. These packages can be installed with the appropriate package manager for your OS:

- SLES: awk environment-modules glibc-devel gzip python3 tar
- RHEL: environment-modules glibc-devel procps python3
- Amazon Linux: environment-modules glibc-devel gzip procps python3 tar
- Ubuntu: environment-modules libc6-dev python3

Note: The minimum supported version for Python is version 3.6.

You must have at least 2 GB of free hard disk space to both download and unpack the Arm® Compiler for Linux package. You must also have an additional 6 GB of free space to install the package.

3.3 Download the product

Arm delivers the files through the Arm Developer website:

https://developer.arm.com/Tools%20and%20Software/Arm%20Compiler%20for%20Linux#Technical-Specifications

3.4 Unpack the product

To unpack the package, extract the tar file contents using a tar utility:

```
tar -xvf <package_name>.tar
```

3.5 Directory structure:

Shows the top-level directory structure of this installer package, which is available after you unpack the bundle:

```
license_terms/
arm-compiler-for-linux-23.04.1*.sh
RELEASE_NOTES.txt
```

3.6 Install the product

To install Arm® Compiler for Linux, navigate into the extracted package directory (<package_name>) and run the installation script as a privileged user. Pass any options to configure the installation:


```
cd path/to/<package_name>/
./arm-compiler-for-linux-23.04.1*.sh [option]...
```

Some common installation options are:

- For a headless installation and to automatically accept the EULA, use the '–accept' option.
- To install to an alternate location to the default, use the '–install-to <install_location>' option.

For a full list of supported installation options pass the '-h' or '–help' options to the installer script.

To learn more about installing Arm® Compiler for Linux, see:

https://developer.arm.com/documentation/102621/0100/Install-Arm-Compiler-for-Linux

The installer displays the EULA and prompts you to agree to the terms. Type 'yes' at the prompt to continue.

All the packages are unpacked to <install_location>/<package_name> with environment modulefiles available under <install_location>/modulefiles. The default installation location is /opt/arm/. Local installs have the same directory structure starting from your chosen installation root.

3.7 RPM and DEB files

The install packages contain RPM (.rpm) files, for Linux distributions that use the Red Hat Package Manager (including SLES and Amazon Linux), or DEB (.deb) files, for Debian-based Linux distributions.

To extract the .rpm or .deb files from the installer, run the installer script with the '-s' or '–save-packages-to <directory_location>' option. If <directory_location> is not an empty directory, you also need to include the '-f' or '–force' option. The installer script requires you to accept the EULA. If you accept the EULA, the .rpm or .deb files extract to <directory_location>.

RPM files are signed by Arm's HPC GPG key. DEB files are not signed. To verify RPM files, you can download and import the Arm's HPC GPG key, and check the signatures:

1. Download the Arm HPC GPG public key from:

   https://developer.arm.com/-/media/files/keys/GPG-PUB-KEY-ARM-HPC-SW-TOOLS.PUB

2. Import the GPG key, run:

   ```
   rpm --import GPG-PUB-KEY-ARM-HPC-SW-TOOLS.PUB
   ```

3. Check the signature of an .rpm file, run:

   ```
   rpm -K <rpm_file>
   ```

To install Arm® Compiler for Linux using rpm/deb files, navigate into the extracted rpm/deb files directory, run:

```
rpm -i <list_of_rpm_files>
dpkg -i <list_of_deb_files>
```

Note: Arm does not recommend that you install directly from the .rpm or .deb files. Only experienced users who are comfortable with this type of installation route should attempt to install the Arm® Compiler for Linux package using this method.

3.8 Run the product

1. Load the environment module:

   - Ensure you have access to modules, replace /opt/arm with

   <install_location> if necessary, and use:


```
module use /opt/arm/modulefiles
module avail
```

- For Arm® C/C++/Fortran Compiler, use:

```
module load acfl/23.04.1
```

To also use Arm® Performance Libraries, include the `-armpl` compiler option when linking your executable. You do not need to load the Arm Performance Libraries modulefile.

- For GCC 12.2.0 only, use:

```
module load gnu/12.2.0
```

- For GCC 12.2.0 with Arm® Performance Libraries, use:

```
module load gnu/12.2.0
module load armpl/23.04.1
```

2. Generate your executable binary.

   To generate an executable binary with Arm® Compiler for Linux, compile your program with Arm® C/C++/Fortran Compiler and specify any options ([options]), the output binary name (-o <binary>), and the input file (<input>):

```
{armclang|armclang++|armflang} [options] -o <binary> <input>
```

   Refer to the GCC documentation to see the equivalent command syntax for the GCC compiler.

3.9 Examples

Example code is included in this suite as part of Arm Performance Libraries. This code can be found at:

```
<install_location>/<ARMPL_Name>*<ARMPL_Version>*/examples*
```

Examples that use, and do not use, SVE are included for each of Arm C/C++/Fortran Compiler and GCC.

Multiple examples directories are provided in the installation. The suffix of the directory name indicates whether the examples inside link to the 32-bit ('_lp64') or 64-bit ('_ilp64') integer variants, and sequential (no suffix indicator) or OpenMP ('_mp') multi-threaded variants, of Arm Performance Libraries.

The default set of examples in the 'examples' directory link to the sequential, 32-bit integers variant of Arm® Performance Libraries.

To build the default set of examples:

- For Arm® Compiler for Linux:

  1. Copy the 'examples' directory somewhere writeable:

```
cp -r <install_location>/armpl-23.04.1*arm-linux-compiler/examples ./
cd examples
```

  2. Load the Arm® Compiler for Linux environment module:

```
module load acfl/23.04.1
```

  3. Build the examples:

```
make
```

- For GCC:

1. Copy the 'examples' directory somewhere writeable:

```
cp -r <install_location>/armpl-23.04.1*gcc/examples ./
cd examples
```

2. Load the GCC environment modules:

```
module load gnu/12.2.0
module load armpl/23.04.1
```

3. Build the examples:

```
make
```

For more information about the Arm® Performance Libraries examples, see:

https://developer.arm.com/documentation/102574/0100/Compile-and-test-the-examples

3.10 Uninstall

For convenience, this package includes an "uninstall.sh" script at:

```
<install_location>/arm-compiler-for-linux-23.04.1*/uninstall.sh
```

This script attempts to uninstall all the components supplied as part of Arm® Compiler for Linux. However, if other packages outside of this product depend on the GCC component, GCC will not be uninstalled. Packages extracted using the –save-packages-to option and installed using rpm/dpkg commands should be uninstalled in the same way and deleted manually.

4 Support

The documentation that is available for Arm® Compiler for Linux can be found on the product resources page on the Arm Developer website:

https://developer.arm.com/Tools%20and%20Software/Arm%20Compiler%20for%20Linux#Resources

You can also find a subset of that documentation, available in

```
<install_location>/<package_name>/share.
```

These deliverables are being released under the terms of the agreement between Arm and each licensee (the “Agreement”). All planned verification and validation is complete. The release is suitable for volume production under the terms of the Agreement.

4.1 OS

This suite is supported on the following Linux platforms:

- AArch64 RHEL 7, 8 and 9
- AArch64 SLES 15
- AArch64 Ubuntu 20.04 and 22.04
- AArch64 Amazon Linux 2 and 2023

Full information about the platforms supported by Arm® Compiler for Linux is available on the Arm Developer website:

https://developer.arm.com/Tools%20and%20Software/Arm%20Compiler%20for%20Linux#Supported-Devices

5 Release history

A full release history (with release notes) for Arm® Compiler for Linux is available on the Arm Developer website:

https://developer.arm.com/documentation/107578/latest

6 Glossary

The Arm Glossary is a list of terms that are 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: https://developer.arm.com/glossary.

