AAPCS stands for the ARM Architecture Procedure Call Standard. It is a set of rules and conventions established by ARM for how software components interact during the procedure call process. The purpose of AAPCS is to ensure compatibility and interoperability between software components compiled with different compilers or written in different programming languages.

AAPCS defines guidelines for the following aspects of procedure calls:

Register usage: AAPCS defines which registers are used for passing function arguments and return values. It specifies how the registers are used based on their size (e.g., 32-bit or 64-bit) and the number of arguments.

Stack frame layout: AAPCS defines the layout of the stack frame, which includes the organization of local variables, saved registers, and return addresses. It specifies rules for stack alignment, parameter passing on the stack, and stack growth direction.

Exception handling: AAPCS provides guidelines for handling exceptions and interrupts, including how to save and restore the necessary context during exception handling.

Interoperability with other languages: AAPCS defines guidelines for interoperability between assembly language and high-level languages, such as C or C++. It ensures that function calls between different languages work correctly by specifying rules for name mangling, calling conventions, and parameter passing.

The usage of AAPCS depends on the specific ARM architecture and the compiler being used. To adhere to AAPCS guidelines, you need to ensure that the following are in place:

Compiler settings: Configure the compiler options to use the appropriate AAPCS variant for your target architecture. Different ARM architectures may have different AAPCS variants, such as AAPCS32 or AAPCS64.

Function declaration and definition: When declaring and defining functions, follow the rules specified by AAPCS for parameter passing, return values, and stack frame layout. Ensure that the correct calling convention is used, and the function signatures match between the caller and the callee.

Assembly language interactions: If you are writing assembly code that interacts with high-level language code, such as C or C++, you need to ensure that the assembly code adheres to the AAPCS guidelines for register usage, stack frame layout, and parameter passing.

By following the AAPCS guidelines, you can ensure that software components compiled with different compilers or written in different languages can work together seamlessly.

It's important to consult the documentation and specifications for the specific ARM architecture and compiler you are using to get the detailed information on the AAPCS variant and its specific rules for that architecture.