CHAPTER 3

Style Guidelines for packetC Program

Introduction to packetC Style Guidelines

This chapter covers packetC coding-style recommendations based on common C++ development community practices. The following recommendations establish the guidelines that CloudShield-developed software is expected to follow within packetC. As always with style guidelines, these are suggestions and individual third-party developers may choose to follow their own style guidelines and packetC compilers shall not be your jury.

While packetC has many traits similar to C and C++, there are also several deviations. As such, C or C++ style guidelines developed elsewhere do not cover all aspects of the packetC grammar to which style applies. This guide is intended to cover areas common to C++ and packetC as well as those areas specific to packetC.

The CloudShield packetC Integrated Development Environment (IDE) provides an editor that improves the readability of code by color-coding, and some automated features for formatting code automatically to conform to the packetC style. These are tools and are considered outside of the guidelines of the style guide.

Meaning of Wording in packetC Style Guidelines

As packetC is a language for networking bearing its roots in representing network protocols, identifying the source for definitions of words significant to developing guidelines in style must be Internet RFCs. The following portions of text are modeled on the best practices defined in RFC 2119.

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described below and have special meaning in this chapter on packetC Style.

- **MUST**—This word, or the terms “REQUIRED” or “SHALL”, means that the definition is an absolute requirement of the style guide.
- **MUST NOT**—This phrase, or the phrase “SHALL NOT”, means that the definition is an absolute prohibition of the style guide.
- **SHOULD**—This word, or the adjective “RECOMMENDED”, means that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.
• **SHOULD NOT**—This phrase, or the phrase “NOT RECOMMENDED”, means that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.

• **MAY**—This word, or the adjective “OPTIONAL”, means that an item is truly optional. One may choose to follow the guideline or not and it will have no effect on whether the code is following the style guidelines.

**Last Clarification**

These guidelines on style for packetC are intended to improve the readability of code as well as help promote a consistent style, making it easier to share code within the packetC community. This guide is not expected to be exhaustive of every scenario and it is expected that the rationale to violate or differ in style from this guide will be hotly debated. Please remember these guidelines are not requirements to a particular coding style forced on the entire packetC development community.

**Naming Conventions for Variables, Types, and Functions**

Throughout this document, four different cases are used for names to help identify the type of a name when seen in code. As packetC, like C, is a case-sensitive language, these styles not only aid in identification of types, but also in assurance of scope benefiting auditing of code. Listed below are the four difference cases leveraged in the style guide:

- **UPPERCASE**—All characters are uppercased.
- **lowercase**—All characters are lowercased.
- **lowerCamelCase**—The first character is lowercased with the first character of each word following capitalized.
- **UpperCamelCase**—The first character is uppercased with the first character of each word following capitalized.

Scoping in packetC is critical to dealing with parallel processing. There are two major scopes that affect naming conventions—namely, global and packet scope. Global scope data are visible to all packets being processed within packetC, while packet scope data are only visible within the processing of the current packet. The third scope, block scope, is a tighter form within packet scope and for naming follows packet scope guidelines. It is important to easily distinguish data that are global and have potential impacts from parallel processing from those data elements that are safe from the impacts of parallel processing. The packetC style guidelines present a method for distinguishing variables from types and functions while also distinguishing variables in different scopes. As always, variables should be declared in as small a scope as possible to protect against conflicts.

**Variables**

Variable names are case sensitive and can only begin with a letter. The rest of the characters can be letters, digits, and underscore (“_”) characters. No white space is permitted in variable names for