**Green Pace Developer: Security Policy Guide Template**



# Green Pace Secure Development Policy

## Contents

[Overview 2](#_Toc52464053)

[Purpose 2](#_Toc52464054)

[Scope 2](#_Toc52464055)

[Module Three Milestone 2](#_Toc52464056)

[Ten Core Security Principles 2](#_Toc52464057)

[C/C++ Ten Coding Standards 3](#_Toc52464058)

[Coding Standard 1 4](#_Toc52464059)

[Coding Standard 2 5](#_Toc52464060)

[Coding Standard 3 6](#_Toc52464061)

[Coding Standard 4 7](#_Toc52464062)

[Coding Standard 5 8](#_Toc52464063)

[Coding Standard 6 9](#_Toc52464064)

[Coding Standard 7 10](#_Toc52464065)

[Coding Standard 8 11](#_Toc52464066)

[Coding Standard 9 13](#_Toc52464067)

[Coding Standard 10 14](#_Toc52464068)

[Defense-in-Depth Illustration 15](#_Toc52464069)

[Project One 15](#_Toc52464070)

[1. Revise the C/C++ Standards 15](#_Toc52464071)

[2. Risk Assessment 15](#_Toc52464072)

[3. Automated Detection 15](#_Toc52464073)

[4. Automation 15](#_Toc52464074)

[5. Summary of Risk Assessments 16](#_Toc52464075)

[6. Create Policies for Encryption and Triple A 16](#_Toc52464076)

[7. Map the Principles 17](#_Toc52464077)

[Audit Controls and Management 18](#_Toc52464078)

[Enforcement 18](#_Toc52464079)

[Exceptions Process 18](#_Toc52464080)

[Distribution 19](#_Toc52464081)

[Policy Change Control 19](#_Toc52464082)

[Policy Version History 19](#_Toc52464083)

[Appendix A Lookups 19](#_Toc52464084)

[Approved C/C++ Language Acronyms 19](#_Toc52464085)

## Overview

Software development at Green Pace requires consistent implementation of secure principles to all developed applications. Consistent approaches and methodologies must be maintained through all policies that are uniformly defined, implemented, governed, and maintained over time.

## Purpose

This policy defines the core security principles; C/C++ coding standards; authorization, authentication, and auditing standards; and data encryption standards. This article explains the differences between policy, standards, principles, and practices (guidelines and procedure): [Understanding the Hierarchy of Principles, Policies, Standards, Procedures, and Guidelines](https://www.linkedin.com/pulse/understanding-hierarchy-principles-policies-standards-wally-beddoe/).

## Scope

This document applies to all staff that create, deploy, or support custom software at Green Pace.

## Module Three Milestone

### Ten Core Security Principles

| **Principles** | Write a short paragraph explaining each of the 10 principles of security. |
| --- | --- |
| 1. ValidateInput Data | Validate all user input before entering it into the program to ensure its accuracy. Additionally, sanitize the input to mitigate potential security risks like SQL injection attacks. |
| 1. Heed Compiler Warnings | Compiler warnings do not prevent the execution of your program. Nevertheless, these warnings may signal a possible issue within the code that could lead to unexpected behavior in the program and potentially highlight a security vulnerability. |
| 1. Architect and Design for Security Policies | It is imperative to take into account security policy from the inception of the development process. The integration of policies into the overarching architecture and design of the application is crucial. This approach will guarantee adherence to the policies across the entire program. |
| 1. Keep It Simple | In essence, this implies prioritizing straightforward code over intricate solutions. The greater the complexity of a program, the higher the likelihood of encountering errors or vulnerabilities that could potentially be exploited. |
| 1. Default Deny | Default denial refers to the automatic denial of access and privileges, which are only granted upon explicit request. This approach effectively restricts the scope of potential problems or security breaches by confining them to a specific area. |
| 1. Adhere to the Principle of Least Privilege | The concept of least privilege asserts that the minimum level of authority required to accomplish a task should be granted. This practice effectively limits the scope of any potential problem to a more restricted domain. |
| 1. Sanitize Data Sent to Other Systems | Prior to transmitting data to another system, it is crucial to verify the validity and format of the data in accordance with the expectations of the receiving system. This practice offers the advantage of mitigating the potential risks associated with the receiving system's failure to adequately validate its input, thereby minimizing the likelihood of any consequential issues arising. |
| 1. Practice Defense in Depth | Defense in depth is a security strategy that involves the implementation of multiple layers of protection. This approach aims to enhance the resilience of a system by introducing redundancy. If one layer of security fails, there are additional layers in place to safeguard the system and maintain its security. |
| 1. Use Effective Quality Assurance Techniques | Thorough testing, coupled with effective quality assurance techniques, plays a crucial role in identifying and resolving any potential issues before they can be exploited. Consequently, this meticulous approach leads to the development of a significantly more secure program. |
| 1. Adopt a Secure Coding Standard | Implementing a secure coding standard is essential for ensuring the consistent delivery of securely coded software, while also ensuring adherence to best practices. However, even the most robust security policy is ineffective if not enforced, underscoring the importance of establishing enforceable requirements to guarantee compliance with the policies. |

### C/C++ Ten Coding Standards

Complete the coding standards portion of the template according to the Module Three milestone requirements. In Project One, follow the instructions to add a layer of security to the existing coding standards. Please start each standard on a new page, as they may take up more than one page. The first seven coding standards are labeled by category. The last three are blank so you may choose three additional standards. Be sure to label them by category and give them a sequential number for that category. Add compliant and noncompliant sections as needed to each coding standard.

#### Coding Standard 1

| **Coding Standard** | **Label** | **Avoid information leakage when passing a class object across a trust boundary** |
| --- | --- | --- |
| **Data Type** | STD-001-CPP | When passing a pointer to a class object instance across a trust boundary to a different trusted domain, the programmer must ensure that the padding bits of such an object do not contain sensitive information. |

| **Noncompliant Code** |
| --- |
| This noncompliant code example runs in kernel space and copies data from arg to user space. However, padding bits may be used within the object, for example, to ensure the proper alignment of class data members |
| |  | | --- | | #include <cstddef>    **struct** test {  **int** a;  **char** b;  **int** c;  };    // Safely copy bytes to user space  **extern** **int** copy\_to\_user(**void** \*dest, **void** \*src, std::**size\_t** size);    **void** do\_stuff(**void** \*usr\_buf) {    test arg{1, 2, 3};    copy\_to\_user(usr\_buf, &arg, **sizeof**(arg));  } | |

| **Compliant Code** |
| --- |
| This compliant solution serializes the structure data before copying it to an untrusted context. |
| #include <cstddef>  #include <cstring>    **struct** test {  **int** a;  **char** b;  **int** c;  };  // Safely copy bytes to user space.  **extern** **int** copy\_to\_user(**void** \*dest, **void** \*src, std::**size\_t** size);    **void** do\_stuff(**void** \*usr\_buf) {    test arg{1, 2, 3};    // May be larger than strictly needed.    unsigned **char** buf[**sizeof**(arg)];    std::**size\_t** offset = 0;      std::**memcpy**(buf + offset, &arg.a, **sizeof**(arg.a));    offset += **sizeof**(arg.a);    std::**memcpy**(buf + offset, &arg.b, **sizeof**(arg.b));    offset += **sizeof**(arg.b);    std::**memcpy**(buf + offset, &arg.c, **sizeof**(arg.c));    offset += **sizeof**(arg.c);      copy\_to\_user(usr\_buf, buf, offset /\* size of info copied \*/);  } |

**Note: Stop here for the milestone. Complete this section for Project One in Module Six.**

| **Principles(s):** This standard maps to Validate Input Data since the class object can contain data that needs to be passed correctly. |
| --- |

**Threat Level**

| **Severity** | **Likelihood** | **Remediation Cost** | **Priority** | **Level** |
| --- | --- | --- | --- | --- |
| Low | Unlikely | High | P1 | L3 |

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| Axivion Bauhaus Suite | 7.2.0 | CertC++ -DCL55 |  |
| CodeSonar | 8.1p0 | MISC.PADDING.POTB | Padding Passed Across A Trust Bound |
| Helix QAC | 2024.1 | DF4941, DF4942, DF4943 |  |
| Parasoft C/C++ Test | 2023.1 | CERT\_CPP-DCL55-a | A pointer to a structure should not be passed to a function that can copy data to the user space |
| Polyspace Bug Finder | R2024a | CERT C++: DCL55-CPP | Checks for information leakage due to structure padding (rule partially covered) |

#### Coding Standard 2

| **Coding Standard** | **Label** | **Ensure that division and remainder operations do not result in divide-by-zero errors** |
| --- | --- | --- |
| **Data Value** | STD-002-CLG | The result of the / operator is the quotient from the division of the first arithmetic operand by the second arithmetic operand. Division operations are susceptible to divide-by-zero errors. Overflow can also occur during two's complement signed integer division when the dividend is equal to the minimum (most negative) value for the signed integer type and the divisor is equal to −1. |

| **Noncompliant Code** |
| --- |
| This noncompliant code example prevents signed integer overflow in compliance with INT32-C. Ensure that operations on signed integers do not result in overflow but fails to prevent a divide-by-zero error during the division of the signed operands s\_a and s\_b |
| #include <limits.h>    **void** func(**signed** **long** s\_a, **signed** **long** s\_b) {  **signed** **long** result;  **if** ((s\_a == LONG\_MIN) && (s\_b == -1)) {      /\* Handle error \*/    } **else** {      result = s\_a / s\_b;    }    /\* ... \*/  } |

| **Compliant Code** |
| --- |
| This compliant solution tests the division operation to guarantee there is no possibility of divide-by-zero errors or signed overflow: |
| #include <limits.h>  **void** func(**signed** **long** s\_a, **signed** **long** s\_b) {  **signed** **long** result;  **if** ((s\_b == 0) || ((s\_a == LONG\_MIN) && (s\_b == -1))) {      /\* Handle error \*/    } **else** {      result = s\_a / s\_b;    }    /\* ... \*/  } |

**Note: Stop here for the milestone. Complete this section for Project One in Module Six.**

| **Principles(s):** This standard follows Validate Input Data because if there is no check for divide-by-zero or for remainder divide-by-zero, then it will crash the program and could create a data leak for anyone to access. |
| --- |

**Threat Level**

| **Severity** | **Likelihood** | **Remediation Cost** | **Priority** | **Level** |
| --- | --- | --- | --- | --- |
| Low | Likely | Medium | P6 | L2 |

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Astrée](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87152428) | 24.04 | **int-division-by-zero**  **int-modulo-by-zero** | Fully checked |
| [Axivion Bauhaus Suite](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=125337650) | 7.2.0 | **CertC-INT33** |  |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/c/CodeSonar) | 8.1p0 | **LANG.ARITH.DIVZERO LANG.ARITH.FDIVZERO** | Division by zero Float Division By Zero |
| [Compass/ROSE](https://wiki.sei.cmu.edu/confluence/display/c/Rose) |  |  | Can detect some violations of this rule (In particular, it ensures that all operations involving division or modulo are preceded by a check ensuring that the second operand is nonzero.) |
| [Coverity](https://wiki.sei.cmu.edu/confluence/display/c/Coverity) | 2017.07 | **DIVIDE\_BY\_ZERO** | Fully implemented |
| [Cppcheck](https://wiki.sei.cmu.edu/confluence/display/c/Cppcheck) | 1.66 | **zerodiv zerodivcond** | Context sensitive analysis of division by zero Not detected for division by struct member / array element / pointer data that is 0 Detected when there is unsafe division by variable before/after test if variable is zero |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/c/Helix+QAC) | 2024.1 | **C2830**  **C++2830**  **DF2831, DF2832, DF2833** |  |
| [Klocwork](https://wiki.sei.cmu.edu/confluence/display/c/Klocwork) | 2024.1 | **DBZ.CONST** **DBZ.CONST.CALL** **DBZ.GENERAL** **DBZ.ITERATOR** **DBZ.ITERATOR.CALL** |  |
| [LDRA tool suite](https://wiki.sei.cmu.edu/confluence/display/c/LDRA) | 9.7.1 | **43 D, 127 D, 248 S, 629 S, 80 X** | Partially implemented |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/c/Parasoft) | 2023.1 | **CERT\_C-INT33-a** | Avoid division by zero |
| [Parasoft Insure++](https://wiki.sei.cmu.edu/confluence/display/c/Parasoft) |  |  | Runtime analysis |
| [Polyspace Bug Finder](https://wiki.sei.cmu.edu/confluence/display/c/Polyspace+Bug+Finder) | R2024a | [CERT C: Rule INT33-C](https://www.mathworks.com/help/bugfinder/ref/certcruleint33c.html) | Checks for:   * Integer division by zero * Tainted division operand * Tainted modulo operand   Rule fully covered. |
| [SonarQube C/C++ Plugin](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87151949) | 3.11 | [**S3518**](https://www.sonarsource.com/products/codeanalyzers/sonarcfamilyforcpp/rules-c.html#RSPEC-3518) |  |
| [PVS-Studio](https://wiki.sei.cmu.edu/confluence/display/c/PVS-Studio) | 7.31 | [**V609**](https://pvs-studio.com/en/docs/warnings/v609/) |  |
| [TrustInSoft Analyzer](https://wiki.sei.cmu.edu/confluence/display/c/TrustInSoft+Analyzer) | 1.38 | **division\_by\_zero** | Exhaustively verified (see [one compliant and one non-compliant example](https://taas.trust-in-soft.com/tsnippet/t/c37797b7)). |

#### Coding Standard 3

| **Coding Standard** | **Label** | **Do not attempt to modify string literals** |
| --- | --- | --- |
| **String Correctness** | STD-003-CLG | Avoid assigning a string literal to a pointer to non-const or casting a string literal to a pointer to non-const. |

| **Noncompliant Code** |
| --- |
| In this noncompliant code example, the char pointer str is initialized to the address of a string literal. |
| **char** \*str  = "string literal";  str[0] = 'S'; |

| **Compliant Code** |
| --- |
| As an array initializer, a string literal specifies the initial values of characters in an array as well as the size of the array. |
| **char** str[] = "string literal";  str[0] = 'S'; |

**Note: Stop here for the milestone. Complete this section for Project One in Module Six.**

| **Principles(s):** This standard maps to Validate Input Data because if a string literal is assigned to or casted to a non-const, then that string literal is said to be able to be changed at any time when it shouldn’t be. |
| --- |

**Threat Level**

| **Severity** | **Likelihood** | **Remediation Cost** | **Priority** | **Level** |
| --- | --- | --- | --- | --- |
| Low | Likely | Low | P9 | L2 |

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Astrée](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87152428) | 24.04 | **string-literal-modfication** **write-to-string-literal** | Fully checked |
| [Axivion Bauhaus Suite](https://wiki.sei.cmu.edu/confluence/display/c/Axivion+Bauhaus+Suite) | 7.2.0 | **CertC-STR30** | Fully implemented |
| [Compass/ROSE](https://wiki.sei.cmu.edu/confluence/display/c/Rose) |  |  | Can detect simple violations of this rule |
| [Coverity](https://wiki.sei.cmu.edu/confluence/display/c/Coverity) | 2017.07 | **PW** | Deprecates conversion from a string literal to "char \*" |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/c/Helix+QAC) | 2024.1 | **C0556, C0752, C0753, C0754**  **C++3063, C++3064, C++3605, C++3606, C++3607** |  |
| [Klocwork](https://wiki.sei.cmu.edu/confluence/display/c/Klocwork) | 2024.1 | **CERT.STR.ARG.CONST\_TO\_NONCONST** **CERT.STR.ASSIGN.CONST\_TO\_NONCONST** |  |
| [LDRA tool suite](https://wiki.sei.cmu.edu/confluence/display/c/LDRA) | 9.7.1 | **157 S** | Partially implemented |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/c/Parasoft) | 2023.1 | **CERT\_C-STR30-a** **CERT\_C-STR30-b** | A string literal shall not be modified Do not modify string literals |
| [PC-lint Plus](https://wiki.sei.cmu.edu/confluence/display/c/PC-lint+Plus) | 1.4 | **489, 1776** | Partially supported |
| [Polyspace Bug Finder](https://wiki.sei.cmu.edu/confluence/display/c/Polyspace+Bug+Finder) | R2024a | [CERT C: Rule STR30-C](https://www.mathworks.com/help/bugfinder/ref/certcrulestr30c.html) | Checks for writing to const qualified object (rule fully covered) |
| [PVS-Studio](https://wiki.sei.cmu.edu/confluence/display/c/PVS-Studio) | 7.31 | [**V675**](https://pvs-studio.com/en/docs/warnings/v675/) |  |
| [RuleChecker](https://wiki.sei.cmu.edu/confluence/display/c/RuleChecker) | 24.04 | **string-literal-modfication** | Partially checked |
| [Splint](https://wiki.sei.cmu.edu/confluence/display/c/Splint) | 3.1.1 |  |  |
| [TrustInSoft Analyzer](https://wiki.sei.cmu.edu/confluence/display/c/TrustInSoft+Analyzer) | 1.38 | mem\_access | Exhaustively verified (see [one compliant and one non-compliant example](https://taas.trust-in-soft.com/tsnippet/t/952d807d)). |

#### Coding Standard 4

| **Coding Standard** | **Label** | **Normalize strings before validating them** |
| --- | --- | --- |
| **SQL Injection** | STD-004-JAV | Applications that accept untrusted input should normalize the input before validating it. Normalization is important because in Unicode, the same string can have many different representations. |

| **Noncompliant Code** |
| --- |
| This noncompliant code example attempts to validate the String before performing normalization. |
| // String s may be user controllable  // \uFE64 is normalized to < and \uFE65 is normalized to > using the NFKC normalization form  String s = "\uFE64" + "script" + "\uFE65";    // Validate  Pattern pattern = Pattern.compile("[<>]"); // Check for angle brackets  Matcher matcher = pattern.matcher(s);  **if** (matcher.find()) {    // Found black listed tag  **throw** **new** IllegalStateException();  } **else** {    // ...  }    // Normalize  s = Normalizer.normalize(s, Form.NFKC); |

| **Compliant Code** |
| --- |
| This compliant solution normalizes the string before validating it. Alternative representations of the string are normalized to the canonical angle brackets. |
| String s = "\uFE64" + "script" + "\uFE65";    // Normalize  s = Normalizer.normalize(s, Form.NFKC);    // Validate  Pattern pattern = Pattern.compile("[<>]");  Matcher matcher = pattern.matcher(s);  **if** (matcher.find()) {    // Found blacklisted tag  **throw** **new** IllegalStateException();  } **else** {    // ...  } |

**Note: Stop here for the milestone. Complete this section for Project One in Module Six.**

| **Principles(s):** This standard relates to Validate Input Data because a single Unicode character can be construed as many different variations. As long as every string is normalized, the program will know what is meant by the strings. |
| --- |

**Threat Level**

| **Severity** | **Likelihood** | **Remediation Cost** | **Priority** | **Level** |
| --- | --- | --- | --- | --- |
| High | Probable | Medium | P12 | L1 |

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [The Checker Framework](https://wiki.sei.cmu.edu/confluence/display/java/The+Checker+Framework) | 2.1.3 | **Tainting Checker** | Trust and security errors (see Chapter 8) |
| [Fortify](https://wiki.sei.cmu.edu/confluence/display/java/Fortify) | 1.0 | **Process\_Control** | Implemented |

#### Coding Standard 5

| **Coding Standard** | **Label** | **Do not access freed memory** |
| --- | --- | --- |
| **Memory Protection** | STD-005-CPP | It is at the memory manager's discretion when to reallocate or recycle the freed memory. When memory is freed, all pointers into it become invalid, and its contents might either be returned to the operating system, making the freed space inaccessible, or remain intact and accessible. |

| **Noncompliant Code** |
| --- |
| In this noncompliant code example, s is dereferenced after it has been deallocated. If this access results in a write-after-free, the vulnerability can be exploited to run arbitrary code with the permissions of the vulnerable process. |
| #include <new>    **struct** S {  **void** f();  };    **void** g() noexcept(**false**) {    S \*s = **new** S;    // ...  **delete** s;    // ...    s->f();  } |

| **Compliant Code** |
| --- |
| In this compliant solution, the dynamically allocated memory is not deallocated until it is no longer required. |
| #include <new>  **struct** S {  **void** f();  };    **void** g() noexcept(**false**) {    S \*s = **new** S;    // ...    s->f();  **delete** s;  } |

**Note: Stop here for the milestone. Complete this section for Project One in Module Six.**

| **Principles(s):** This standard maps to Practice Defense In Depth because if someone is trying to access memory that is freed, then that memory should not contain anything from the program since it should have been cleansed and returned to the system. |
| --- |

**Threat Level**

| **Severity** | **Likelihood** | **Remediation Cost** | **Priority** | **Level** |
| --- | --- | --- | --- | --- |
| High | Likely | Medium | P18 | L1 |

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Astrée](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=222953724) | 22.10 | **dangling\_pointer\_use** |  |
| [Axivion Bauhaus Suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Axivion+Bauhaus+Suite) | 7.2.0 | **CertC++-MEM50** |  |
| [Clang](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Clang) | 3.9 | clang-analyzer-cplusplus.NewDelete clang-analyzer-alpha.security.ArrayBoundV2 | Checked by clang-tidy, but does not catch all violations of this rule. |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/cplusplus/CodeSonar) | 8.1p0 | **ALLOC.UAF** | Use after free |
| [Compass/ROSE](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Rose) |  |  |  |
| [Coverity](https://wiki.sei.cmu.edu/confluence/display/c/Coverity) | v7.5.0 | **USE\_AFTER\_FREE** | Can detect the specific instances where memory is deallocated more than once or read/written to the target of a freed pointer |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2024.1 | **C++4303, C++4304** |  |
| [Klocwork](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Klocwork) | 2024.1 | **UFM.DEREF.MIGHT** **UFM.DEREF.MUST** **UFM.FFM.MIGHT** **UFM.FFM.MUST** **UFM.RETURN.MIGHT** **UFM.RETURN.MUST** **UFM.USE.MIGHT** **UFM.USE.MUST** |  |
| [LDRA tool suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/LDRA) | 9.7.1 | **483 S, 484 S** | Partially implemented |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2023.1 | **CERT\_CPP-MEM50-a** | Do not use resources that have been freed |
| [Parasoft Insure++](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) |  |  | Runtime detection |
| [Polyspace Bug Finder](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Polyspace+Bug+Finder) | R2024a | [CERT C++: MEM50-CPP](https://www.mathworks.com/help/bugfinder/ref/certcmem50cpp.html) | Checks for:   * Pointer access out of bounds * Deallocation of previously deallocated pointer * Use of previously freed pointer   Rule partially covered. |
| [PVS-Studio](https://wiki.sei.cmu.edu/confluence/display/cplusplus/PVS-Studio) | 7.31 | [**V586**](https://pvs-studio.com/en/docs/warnings/v586/), [**V774**](https://pvs-studio.com/en/docs/warnings/v774/) |  |
| [Splint](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Splint) | 5.0 |  |  |

#### Coding Standard 6

| **Coding Standard** | **Label** | **Do not perform assignments in selection statements** |
| --- | --- | --- |
| **Assertions** | STD-006-CLG | Do not use the assignment operator in the contexts listed in the following table because doing so typically indicates programmer error and can result in unexpected behavior. |

| **Noncompliant Code** |
| --- |
| In this noncompliant code example, an assignment expression is the outermost expression in an if statement: |
| **if** (a = b) {    /\* ... \*/  } |

| **Compliant Code** |
| --- |
| When the assignment of b to a is not intended, the conditional block is now executed when a is equal to b: |
| **if** (a == b) {    /\* ... \*/  } |

**Note: Stop here for the milestone. Complete this section for Project One in Module Six.**

| **Principles(s):** This standard maps to Keep It Simple because it is a simple error that is relatively easy to fix because of an equal sign missing. Where it can become complex is if it is in a multi if statement that might have it hidden. |
| --- |

**Threat Level**

| **Severity** | **Likelihood** | **Remediation Cost** | **Priority** | **Level** |
| --- | --- | --- | --- | --- |
| Low | Likely | Medium | P6 | L2 |

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Astrée](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87152428) | 24.04 | **assignment-conditional** | Fully checked |
| [Axivion Bauhaus Suite](https://wiki.sei.cmu.edu/confluence/display/c/Axivion+Bauhaus+Suite) | 7.2.0 | **CertC-EXP45** |  |
| [Clang](https://wiki.sei.cmu.edu/confluence/display/c/Clang) | 3.9 | -Wparentheses | Can detect some instances of this rule, but does not detect all |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/c/CodeSonar) | 8.1p0 | **LANG.STRUCT.CONDASSIG LANG.STRUCT.SE.COND LANG.STRUCT.USEASSIGN** | Assignment in conditional Condition contains side effects Assignment result in expression |
| [Compass/ROSE](https://wiki.sei.cmu.edu/confluence/display/c/Rose) |  |  | Could detect violations of this recommendation by identifying any assignment expression as the top-level expression in an if or while statement |
| [ECLAIR](https://wiki.sei.cmu.edu/confluence/display/c/ECLAIR) | 1.2 | **CC2.EXP18 CC2.EXP21** | Fully implemented |
| [GCC](https://wiki.sei.cmu.edu/confluence/display/c/GCC) | 4.3.5 |  | Can detect violations of this recommendation when the -Wall flag is used |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/c/Helix+QAC) | 2024.1 | **C3314, C3326, C3344, C3416**  **C++4071, C++4074** |  |
| [Klocwork](https://wiki.sei.cmu.edu/confluence/display/c/Klocwork) | 2024.1 | **ASSIGCOND.CALL** **ASSIGCOND.GEN MISRA.ASSIGN.COND** |  |
| [LDRA tool suite](https://wiki.sei.cmu.edu/confluence/display/c/LDRA) | 9.7.1 | **114 S, 132 S** | Enhanced Enforcement |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/c/Parasoft) | 2023.1 | **CERT\_C-EXP45-b** **CERT\_C-EXP45-d** | Assignment operators shall not be used in conditions without brackets Assignment operators shall not be used in expressions that yield a Boolean value |
| [PC-lint Plus](https://wiki.sei.cmu.edu/confluence/display/c/PC-lint+Plus) | 1.4 | **720** | Partially supported: reports Boolean test of unparenthesized assignment |
| [Polyspace Bug Finder](https://wiki.sei.cmu.edu/confluence/display/c/Polyspace+Bug+Finder) | R2024a | [CERT C: Rule EXP45-C](https://www.mathworks.com/help/bugfinder/ref/certcruleexp45c.html) | Checks for invalid use of = (assignment) operator (rule fully covered) |
| [PVS-Studio](https://wiki.sei.cmu.edu/confluence/display/c/PVS-Studio) | 7.31 | [**V559**](https://pvs-studio.com/en/docs/warnings/v559/), [**V633**](https://pvs-studio.com/en/docs/warnings/v633/), [**V699**](https://pvs-studio.com/en/docs/warnings/v699/) |  |
| [RuleChecker](https://wiki.sei.cmu.edu/confluence/display/c/RuleChecker) | 24.04 | **assignment-conditional** | Fully checked |
| [SonarQube C/C++ Plugin](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87151949) | 3.11 | [**AssignmentInSubExpression**](https://www.sonarsource.com/products/codeanalyzers/sonarcfamilyforcpp/rules-c.html#RSPEC-1121) |  |

#### Coding Standard 7

| **Coding Standard** | **Label** | **Do not leak resources when handling exceptions** |
| --- | --- | --- |
| **Exceptions** | STD-007-CPP | Resources must not be leaked because of throwing an exception, including during the construction of an object. |

| **Noncompliant Code** |
| --- |
| In this noncompliant code example, pst is not properly released when process\_item throws an exception, causing a resource leak. |
| #include <new>    **struct** SomeType {    SomeType() noexcept; // Performs nontrivial initialization.    ~SomeType(); // Performs nontrivial finalization.  **void** process\_item() noexcept(**false**);  };    **void** f() {    SomeType \*pst = **new** (std::**nothrow**) SomeType();  **if** (!pst) {      // Handle error  **return**;    }    **try** {      pst->process\_item();    } **catch** (...) {      // Process error, but do not recover from it; rethrow.  **throw**;    }  **delete** pst;  } |

| **Compliant Code** |
| --- |
| In this compliant solution, the exception handler frees pst by calling delete. |
| #include <new>  **struct** SomeType {    SomeType() noexcept; // Performs nontrivial initialization.    ~SomeType(); // Performs nontrivial finalization.    **void** process\_item() noexcept(**false**);  };    **void** f() {    SomeType \*pst = **new** (std::**nothrow**) SomeType();  **if** (!pst) {      // Handle error  **return**;    }  **try** {      pst->process\_item();    } **catch** (...) {      // Process error, but do not recover from it; rethrow.  **delete** pst;  **throw**;    }  **delete** pst;  } |

**Note: Stop here for the milestone. Complete this section for Project One in Module Six.**

| **Principles(s):** This standard maps to Practice Defense In Depth since exceptions can be used to force access to the system’s data storage. As long as the program can catch the exception, then the resources won’t be leaked through a console or the application window. |
| --- |

**Threat Level**

| **Severity** | **Likelihood** | **Remediation Cost** | **Priority** | **Level** |
| --- | --- | --- | --- | --- |
| Low | Probable | High | P2 | L3 |

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/cplusplus/CodeSonar) | 8.1p0 | **ALLOC.LEAK** | Leak |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2024.1 | **DF4756, DF4757, DF4758** |  |
| [Klocwork](https://www.securecoding.cert.org/confluence/display/cplusplus/Klocwork) | 2024.1 | **CL.MLK** **MLK.MIGHT** **MLK.MUST** **MLK.RET.MIGHT** **MLK.RET.MUST** **RH.LEAK** |  |
| [LDRA tool suite](https://wiki.sei.cmu.edu/confluence/display/cplusplus/LDRA) | 9.7.1 | **50 D** | Partially implemented |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2023.1 | **CERT\_CPP-ERR57-a** | Ensure resources are freed |
| [Polyspace Bug Finder](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Polyspace+Bug+Finder) | R2024a | [CERT C++: ERR57-CPP](https://www.mathworks.com/help/bugfinder/ref/certcerr57cpp.html) | Checks for:   * Resource leak caused by exception * Object left in partially initialized state   Bad allocation in constructor |

#### Coding Standard 8

| **Coding Standard** | **Label** | **Do not destroy a mutex while it is locked** |
| --- | --- | --- |
| Concurrency | STD-008-CLG | Mutex objects are used to protect shared data from being concurrently accessed. If a mutex object is destroyed while a thread is blocked waiting for the lock, critical sections and shared data are no longer protected. |

| **Noncompliant Code** |
| --- |
| This noncompliant code example creates several threads that each invoke the do\_work() function, passing a unique number as an ID. |
| #include <mutex>  #include <thread>    **const** **size\_t** maxThreads = 10;    **void** do\_work(**size\_t** i, std::mutex \*pm) {    std::lock\_guard<std::mutex> lk(\*pm);      // Access data protected by the lock.  }    **void** start\_threads() {    std::**thread** threads[maxThreads];    std::mutex m;    **for** (**size\_t** i = 0; i < maxThreads; ++i) {      threads[i] = std::**thread**(do\_work, i, &m);    }  } |

| **Compliant Code** |
| --- |
| This compliant solution eliminates the race condition by extending the lifetime of the mutex. |
| #include <mutex>  #include <thread>    **const** **size\_t** maxThreads = 10;    **void** do\_work(**size\_t** i, std::mutex \*pm) {    std::lock\_guard<std::mutex> lk(\*pm);      // Access data protected by the lock.  }    std::mutex m;    **void** start\_threads() {    std::**thread** threads[maxThreads];    **for** (**size\_t** i = 0; i < maxThreads; ++i) {      threads[i] = std::**thread**(do\_work, i, &m);    }  } |

**Note: Stop here for the milestone. Complete this section for Project One in Module Six.**

| **Principles(s):** This standard maps to Default Deny because if there is data locked by the system, then the system should be the one and only to delete or destroy that type of data being used at the time. |
| --- |

**Threat Level**

| **Severity** | **Likelihood** | **Remediation Cost** | **Priority** | **Level** |
| --- | --- | --- | --- | --- |
| Medium | Probable | High | P4 | L3 |

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Astrée](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=87152428) | 24.04 |  | Supported, but no explicit checker |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/c/CodeSonar) | 8.1p0 | **CONCURRENCY.LOCALARG** | Local Variable Passed to Thread |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/c/Helix+QAC) | 2024.1 | **DF4961, DF4962** |  |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/c/Parasoft) | 2023.1 | **CERT\_C-CON31-a** **CERT\_C-CON31-b** **CERT\_C-CON31-c** | Do not destroy another thread's mutex Do not use resources that have been freed Do not free resources using invalid pointers |
| [Polyspace Bug Finder](https://wiki.sei.cmu.edu/confluence/display/c/Polyspace+Bug+Finder) | R2024a | [CERT C: Rule CON31-C](https://www.mathworks.com/help/bugfinder/ref/certcrulecon31c.html) | Checks for destruction of locked mutex (rule fully covered) |

#### Coding Standard 9

| **Coding Standard** | **Label** | **Use valid iterator ranges** |
| --- | --- | --- |
| Containers | STD-009-CPP | When iterating over elements of a container, the iterators used must iterate over a valid range. An iterator range is a pair of iterators that refer to the first and past-the-end elements of the range respectively. |

| **Noncompliant Code** |
| --- |
| In this noncompliant example, the two iterators that delimit the range point into the same container, but the first iterator does not precede the second. On each iteration of its internal loop, std::for\_each() compares the first iterator (after incrementing it) with the second for equality; as long as they are not equal, it will continue to increment the first iterator. |
| #include <algorithm>  #include <iostream>  #include <vector>    **void** f(**const** std::vector<**int**> &c) {    std::for\_each(c.end(), c.begin(), [](**int** i) { std::cout << i; });  } |

| **Compliant Code** |
| --- |
| In this compliant solution, the iterator values passed to std::for\_each() are passed in the proper order. |
| #include <algorithm>  #include <iostream>  #include <vector>    **void** f(**const** std::vector<**int**> &c) {    std::for\_each(c.begin(), c.end(), [](**int** i) { std::cout << i; });  } |

**Note: Stop here for the milestone. Complete this section for Project One in Module Six.**

| **Principles(s):** This standard uses Validate Input Data since mixing the begin and end points can cause an error in the program because it would try to go backwards until the system realizes that it is not iterating right. |
| --- |

**Threat Level**

| **Severity** | **Likelihood** | **Remediation Cost** | **Priority** | **Level** |
| --- | --- | --- | --- | --- |
| High | Probable | High | P6 | L2 |

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [Astrée](https://wiki.sei.cmu.edu/confluence/pages/viewpage.action?pageId=222953724) | 22.10 | **overflow\_upon\_dereference** |  |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/cplusplus/CodeSonar) | 8.1p0 | **LANG.MEM.BO** | Buffer Overrun |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2024.1 | **C++3802** |  |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2023.1 | **CERT\_CPP-CTR53-a** **CERT\_CPP-CTR53-b** | Do not use an iterator range that isn't really a range Do not compare iterators from different containers |
| [Polyspace Bug Finder](https://wiki.sei.cmu.edu/confluence/display/c/Polyspace+Bug+Finder) | R2024a | [CERT C++: CTR53-CPP](https://www.mathworks.com/help/bugfinder/ref/certcctr53cpp.html) | Checks for invalid iterator range (rule partially covered). |
| [PVS-Studio](https://wiki.sei.cmu.edu/confluence/display/cplusplus/PVS-Studio) | 7.31 | [**V539**](https://pvs-studio.com/en/docs/warnings/v539/), [**V662**](https://pvs-studio.com/en/docs/warnings/v662/), [**V789**](https://pvs-studio.com/en/docs/warnings/v789/) |  |

#### Coding Standard 10

| **Coding Standard** | **Label** | **Close files when they are no longer needed** |
| --- | --- | --- |
| Input/Output | STD-010-CPP | A call to the std::basic\_filebuf<T>::open() function must be matched with a call to std::basic\_filebuf<T>::close() before the lifetime of the last pointer that stores the return value of the call has ended or before normal program termination, whichever occurs first. |

| **Noncompliant Code** |
| --- |
| In this noncompliant code example, a std::fstream object file is constructed. The constructor for std::fstream calls std::basic\_filebuf<T>::open(), and the default std::terminate\_handler called by std::terminate() is std::abort(), which does not call destructors. |
| #include <exception>  #include <fstream>  #include <string>    **void** f(**const** std::string &fileName) {    std::fstream file(fileName);  **if** (!file.is\_open()) {      // Handle error  **return**;    }    // ...    std::terminate();  } |

| **Compliant Code** |
| --- |
| In this compliant solution, std::fstream::close() is called before std::terminate() is called, ensuring that the file resources are properly closed. |
| #include <exception>  #include <fstream>  #include <string>    **void** f(**const** std::string &fileName) {    std::fstream file(fileName);  **if** (!file.is\_open()) {      // Handle error  **return**;    }    // ...    file.close();  **if** (file.fail()) {      // Handle error    }    std::terminate();  } |

**Note: Stop here for the milestone. Complete this section for Project One in Module Six.**

| **Principles(s):** This standard maps to Default Deny because as soon as the file is done being used, it needs to be closed so no data can be corrupted while the program is still running. |
| --- |

**Threat Level**

| **Severity** | **Likelihood** | **Remediation Cost** | **Priority** | **Level** |
| --- | --- | --- | --- | --- |
| Medium | Unlikely | Medium | P4 | L3 |

**Automation**

| **Tool** | **Version** | **Checker** | **Description Tool** |
| --- | --- | --- | --- |
| [CodeSonar](https://wiki.sei.cmu.edu/confluence/display/cplusplus/CodeSonar) | 8.1p0 | **ALLOC.LEAK** | Leak |
| [Helix QAC](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Helix+QAC) | 2024.1 | **DF4786, DF4787, DF4788** |  |
| [Klocwork](https://www.securecoding.cert.org/confluence/display/cplusplus/Klocwork) | 2024.1 | **RH.LEAK** |  |
| [Parasoft C/C++test](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) | 2023.1 | **CERT\_CPP-FIO51-a** | Ensure resources are freed |
| [Parasoft Insure++](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Parasoft) |  |  | Runtime detection |
| [Polyspace Bug Finder](https://wiki.sei.cmu.edu/confluence/display/cplusplus/Polyspace+Bug+Finder) | R2024a | [CERT C++: FIO51-CPP](https://www.mathworks.com/help/bugfinder/ref/certcfio51cpp.html) | Checks for resource leak (rule partially covered) |

### Defense-in-Depth Illustration

This illustration provides a visual representation of the defense-in-depth best practice of layered security.



## Project One

There are seven steps outlined below that align with the elements you will be graded on in the accompanying rubric. When you complete these steps, you will have finished the security policy.

### Revise the C/C++ Standards

You completed one of these tables for each of your standards in the Module Three milestone. In Project One, add revisions to improve the explanation and examples as needed. Add rows to accommodate additional examples of compliant and noncompliant code. Coding standards begin on the security policy.

### Risk Assessment

Complete this section on the coding standards tables. Enter high, medium, or low for each of the headers, then rate it overall using a scale from 1 to 5, 5 being the greatest threat. You will address each of the seven policy standards. Fill in the columns of severity, likelihood, remediation cost, priority, and level using the values provided in the appendix.

### Automated Detection

Complete this section of each table on the coding standards to show the tools that may be used to detect issues. Provide the tool name, version, checker, and description. List one or more tools that can automatically detect this issue and its version number, name of the rule or check (preferably with link), and any relevant comments or description—if any. This table ties to a specific C++ coding standard.

### Automation

Provide a written explanation using the image provided.



Automation will be used for the enforcement of and compliance to the standards defined in this policy. Green Pace already has a well-established DevOps process and infrastructure. Define guidance on where and how to modify the existing DevOps process to automate enforcement of the standards in this policy. Use the DevSecOps diagram and provide an explanation using that diagram as context.

**[Insert your written explanations here.]**

The main difference between the current DevOps process and the new DevSecOps process will be adding security testing tools to the current set of CI/CD tools being used by the team to ensure security is being involved from the start. While transitioning, Green Pace needs to choose the right tools that are relevant to the code to satisfy use cases now and in the future, involve the security team, prioritize quality over speed to ensure everything is secured, and ensuring code is always monitored.

### Summary of Risk Assessments

Consolidate all risk assessments into one table including both coding and systems standards, ordered by standard number.

| Rule | Severity | Likelihood | Remediation Cost | Priority | Level |
| --- | --- | --- | --- | --- | --- |
| CON31-C | Medium | Probable | High | P4 | L3 |
| CTR53-CPP | High | Probable | High | P6 | L2 |
| DC55-CPP | Low | Unlikely | High | P1 | L3 |
| ERR57-CPP | Low | Probable | High | P2 | L3 |
| EXP45-C | Low | Likely | Medium | P6 | L2 |
| FIO51-CPP | Medium | Unlikely | Medium | P4 | L3 |
| IDS01-J | High | Probable | Medium | P12 | L1 |
| INT33-C | Low | Likely | Medium | P6 | L2 |
| MEM50-CPP | High | Likely | Medium | P18 | L1 |
| STR30-C | Low | Likely | Low | P9 | L2 |

### Create Policies for Encryption and Triple A

Include all three types of encryption (in flight, at rest, and in use) and each of the three elements of the Triple-A framework using the tables provided***.***

* 1. Explain each type of encryption, how it is used, and why and when the policy applies.
  2. Explain each type of Triple-A framework strategy, how it is used, and why and when the policy applies.

**Write policies for each and explain what it is, how it should be applied in practice, and why it should be used**.

| 1. **Encryption** | **Explain what it is and how and why the policy applies.** |
| --- | --- |
| Encryption at rest | Encryption at rest involves data being stored on a disk from being accessed by unauthorized users. This is important because as long as all data is encrypted while at rest, then bad actors can not see the data without the decryption key. |
| Encryption in flight | Encryption in flight is a process that protects data as it moves throughout the network. This is helpful to Green Pace since most associates will not have a direct line to the database and must move through the network to send and receive data. |
| Encryption in use | Encryption in use is a practice that encrypts data being accessed, updated, read, or inputted. Encryption in use is helpful to Green Pace because anyone accessing data in the database should have an encrypted connection to ensure no data can be leaked while accessing information. |

| 1. **Triple-A Framework\*** | **Explain what it is and how and why the policy applies.** |
| --- | --- |
| Authentication | Authentication needs to be used at all times to ensure each person accessing the system is accessing information at their user level and not something that should only be accessed by someone higher up in the company. |
| Authorization | Authorization needs to be used each time a user tries to access a file to ensure the file is allowed to be accessed at their user level, while ensuring the user’s login is logged in the system as having accessed the files. |
| Accounting | Accounting is needed due to keeping track of who is changing the database each time, accessing files, and adding new users to ensure that security is staying a high priority. |

**\***Use this checklist for the Triple A to be sure you include these elements in your policy:

* User logins
* Changes to the database
* Addition of new users
* User level of access
* Files accessed by users

### Map the Principles

Map the principles to each of the standards, and provide a justification for the connection between the two. In the Module Three milestone, you added definitions for each of the 10 principles provided. Now it’s time to connect the standards to principles to show how they are supported by principles. You may have more than one principle for each standard, and the principles may be used more than once. Principles are numbered 1 through 10. You will list the number or numbers that apply to each standard, then explain how each of these principles supports the standard. This exercise demonstrates that you have based your security policy on widely accepted principles. Linking principles to standards is a best practice.

**NOTE:** Green Pace has already successfully implemented the following:

* Operating system logs
* Firewall logs
* Anti-malware logs

The only item you must complete beyond this point is the Policy Version History table.

## Audit Controls and Management

Every software development effort must be able to provide evidence of compliance for each software deployed into any Green Pace managed environment.

Evidence will include the following:

* Code compliance to standards
* Well-documented access-control strategies, with sampled evidence of compliance
* Well-documented data-control standards defining the expected security posture of data at rest, in flight, and in use
* Historical evidence of sustained practice (emails, logs, audits, meeting notes)

## Enforcement

The office of the chief information security officer (OCISO) will enforce awareness and compliance of this policy, producing reports for the risk management committee (RMC) to review monthly. Every system deployed in any environment operated by Green Pace is expected to be in compliance with this policy at all times.

Staff members, consultants, or employees found in violation of this policy will be subject to disciplinary action, up to and including termination.

## Exceptions Process

Any exception to the standards in this policy must be requested in writing with the following information:

* Business or technical rationale
* Risk impact analysis
* Risk mitigation analysis
* Plan to come into compliance
* Date for when the plan to come into compliance will be completed

Approval for any exception must be granted by chief information officer (CIO) and the chief information security officer (CISO) or their appointed delegates of officer level.

Exceptions will remain on file with the office of the CISO, which will administer and govern compliance.

## Distribution

This policy is to be distributed to all Green Pace IT staff annually. All IT staff will need to certify acceptance and awareness of this policy annually.

## Policy Change Control

This policy will be automatically reviewed annually, no later than 365 days from the last revision date. Further, it will be reviewed in response to regulatory or compliance changes, and on demand as determined by the OCISO.

## Policy Version History

| Version | Date | Description | Edited By | Approved By |
| --- | --- | --- | --- | --- |
| 1.0 | 08/05/2020 | Initial Template | David Buksbaum |  |
| 2.0 | 05/26/2024 | Initial Information Entered | Treyton Davis | [Insert text.] |
| 2.1 | 06/16/2024 | New Information Entered | Treyton Davis | [Insert text.] |

## Appendix A Lookups

### Approved C/C++ Language Acronyms

| Language | Acronym |
| --- | --- |
| C++ | CPP |
| C | CLG |
| Java | JAV |