All VB .NET software developed for Optimized Technical Solutions, LLC shall conform to the following standards to ensure quality control, minimize defects, support maintainability, and optimize performance. Software developed for others by Optimized Technical Solutions, LLC will conform to this standard unless another standard is specified.

1. Option Explicit and Option Strict are turned on at the project level.
2. Project level assembly information is populated correctly.
3. A neutral language is specified.
4. Project revision numbers are reviewed and modified as necessary. Versions that are not backward compatible will increment either the major or minor revision number.
5. The project is checked for unused references.
6. Namespace definitions are reviewed.
7. All declarations are reviewed for scoping (Protected, Public, Private, Friend).
8. Code analysis will be performed using FxCop 10.0. All exceptions are approved by the review team.
9. Test lists will be generated for each class and public methods. Tests will be designed to test each branch (If and Case blocks) of the underlying code. All tests will pass successfully prior to publication.
10. A resource file is used for string constants to facilitate localization.
11. Icons and associated resources are located in the resource file and all forms and custom controls are ass assigned an icon.
12. Comments are reviewed for clarity and content as well as spelling and grammar.
13. No TODO: items in the project.
14. There are no active errors or warnings in the project.
15. All projects that contain classes shall have a class diagram generated.
16. Each file has a header block
    - 16.1. Purpose of the file
    - 16.2. Developed by
    - 16.3. Copyright information
        - 16.3.1. Owner
        - 16.3.2. Date

16.4. Revision history
16.4.1. Date
16.4.2. Applicable version
16.4.3. Author
16.4.4. Summary of changes

17. XML help comments are applied to all public, friend, and protected:
    17.1. Interfaces
    17.2. Classes
    17.3. Enumerators
    17.4. Events
    17.5. Properties
    17.6. Functions and subroutines
18. Regions are defined for all classes:
    18.1. Enumerators
    18.2. Event declarations
    18.3. Constants
    18.4. Structures
    18.5. Local variables (fields)
    18.6. Properties
    18.7. Functions and subroutines
19. Each document in a project will include:
    19.1. A header with a description of the document is at the top of each file. The header will include:
        19.1.1. The company name
        19.1.2. Copyright information and date
        19.1.3. A description of the file contents and purpose
        19.1.4. A revision history for each published version
    19.2. Any public enumerators are listed at the beginning of the document and are listed alphabetically

19.3. Any interfaces are listed next alphabetically.

19.4. Any classes are listed next and are listed alphabetically. Within a class, the code will be arranged with:

19.4.1. Nested enumerators are listed first and arranged alphabetically

19.4.2. Events are listed next and arranged alphabetically

19.4.3. Any constants are listed next and are arranged alphabetically

19.4.4. Any structure definitions are listed next and are arranged alphabetically

19.4.5. Local variables (fields) are listed next and are arranged alphabetically

19.4.6. Properties are listed next and are arranged alphabetically

19.4.7. Functions and subroutines are listed and are arranged alphabetically

19.4.7.1. Within functions and subroutines, locally scoped variables may be defined anywhere within the routine to optimize memory allocation. When multiple variables are defined in the same area, they should be listed alphabetically.

19.5. Modules are listed next and are arranged alphabetically. Within a module, the code will be arranged with:

19.5.1. Enumerators are listed first and arranged alphabetically

19.5.2. Any constants are listed next and are arranged alphabetically

19.5.3. Any structure definitions are listed next and are arranged alphabetically

19.5.4. Module level variables are listed next and are arranged alphabetically

19.5.5. Functions and subroutines are listed and are arranged alphabetically

19.5.5.1. Within functions and subroutines, locally scoped variables may be defined anywhere within the routine to optimize memory allocation. When multiple variables are defined in the same area, they should be listed alphabetically.

20. Custom controls include attributes for public properties:

20.1. Browsable

20.2. Description

20.3. Category

20.4. Property editors exist for all collections and objects exposed by the control.

21. Classes that utilize a Dispose method (such as GDI+ graphics) will implement the IDispose interface and will utilize an overloaded Dispose method to free up disposable resources.
22. Source code for each published version is archived in a repository.