New York State Technology Enterprise Corporation

NYSTEC Response to SysTest Request for Interpretation of VVSG Vol. 1 Requirement 5.2.3a, VVSG Vol. II 5.4.2 & NYS 6209 Requirements related to Conditionally Compiled Code

For

New York State Board of Elections

Submitted to:

New York State Board of Elections
40 Steuben Place, Albany NY 12207

February 20, 2009
Version 1

Table of Contents

1. BACKGROUND

NYSTEC has prepared this document to respond to the RFI from SysTest dated 10/03/2008 concerning conditionally compiled software code and how the VVSG as well as NYS Election Law and Regulations prescribe the treatment of such code during source code review and the Trusted Build Process. SysTest requested that the NY State Board of Elections decide upon the following interpretations:

1) Vendors cannot submit any code that will not be compiled or otherwise used in the production compilation of the Trusted Build.
2) Vendors can submit code that is not compiled or otherwise used in the Production Build. This is generally termed "conditionally compiled", which means code that may or may not be ignored during compilation, depending on the compiler options or other environmental variables during the system build.

2. DISCUSSION OF APPLICABLE NYS REGULATIONS AND VVSG REQUIREMENTS

NYS 6209 – Software cannot contain code that could cause improper functioning

> 6209.2.G
> *G. Any submitted voting system's software shall not contain any code, procedures or other material which may disable, disarm or otherwise affect in any manner, the proper operation of the voting system, or which may damage the voting system, any hardware, or any computer system or other property of the State Board or county board, including but not limited to 'viruses', 'worms', 'time bombs', and 'drop dead' devices that may cause the voting system to cease functioning properly at a future time.*

NYSTEC believes that this requirement is violated by the inclusion of any conditionally complied code in the source files used for the Trusted Build, which if compiled could:

- Somehow bypass security designed into the voting system
- Disrupt the voting process
- Load software from an unauthorized source.

Due to the nature of conditionally compiled code (especially the #ifdef preprocessor directive in the C language), it is difficult to have complete assurance whether or not such code gets compiled as part of a build. It is quite easy for a developer or person doing the compile to make a mistake and compile code not meant for the production build into a build. Furthermore, a malicious ITA or voting system employee could easily set the complier option(s) in a manner that is not easily detectible or that would not be flagged in a hash check that could result in a back door or other security breach. The best resolution to this threat is to remove all conditionally compiled code that could alter the functional behavior in any of the ways listed above.

VVSG Vol 1 5.2.3 – Modules that include conditionally compiled code could violate several of the modularity rules for code in VVSG Vol 1 5.2.3. Specifically:

5.2.3.a *Each module shall have a specific function that can be tested and verified independently of the remainder of the code. In practice, some additional modules (such as library modules) may be needed to compile the module under test, but the modular construction allows the supporting modules to be replaced by special test versions that support test objectives.*

NYSTEC believes that the inclusion of conditionally compiled code which, if enabled, could alter the functioning of the module violates the above requirements.

Additionally, 5.2.3.d states that modules should be easy to follow and understand:

5.2.3.d A module is small enough to be easy to follow and understand. Program logic visible on a single page is easy to follow and correct. Volume II, Section 5 provides testing guidelines for the accredited test lab to identify large modules subject to review under this requirement.

NYSTEC believes that the inclusion of conditionally compiled code could violate the above requirement if the code becomes difficult to follow and understand.

3. NYSTEC RECOMMENDATION

NYSTEC believes that SysTest's Option 2 should be the interpretation of the requirements, although NYSTEC has a slightly different process for handling conditionally compiled code in the code review. Testing should ensure that source code review uncovers **any and all** conditionally compiled code.

The code cannot be reliably reviewed if the reviewer does not understand whether a piece of code will be included in the build process that will be used in the Trusted Build. Therefore, the build process defined by the vendor for the Trusted Build must be completely understood by both the persons doing the production compile and by the persons doing the code review. They must agree how the build process will control the compile for items that will be included in the compile (header files, source code files, static libraries, dynamically linked libraries, and any other source that will become part of the final installation set). Depending on the technology, this could mean they must at a minimum:

- Understand all options for the Integrated Development Environment (IDE)
- Understand all Make files,
- Understand the function of all compiler flags, linker flags
- Understand any other pieces included in the build (lexical parsers, or other external pieces that take input from source files and is used in the final installation file set).
- Understand any and all scripts used in the build process

Once the list of input files have been agreed upon by the production compile team and the code reviewers, they must be sure that they are looking at the same versions of all input files.

Then, as part of their code review, reviewers must look at **all** conditionally compiled code. If the code will be compiled in the Trusted Build, it must be reviewed as "regular" code, per the code review procedures. If the code will not be in the Trusted Build, then the reviewer must decide if the conditional code will be permitted by determining if it either:

1) Creates functionality that has the possibility to disrupt voting or the possibility of compromising security (that is, would bypass or override any logical or physical security control of the device). Because of the risk of being included in the Trusted Build (accidentally or maliciously), any code like this that is found must generate a discrepancy.
2) Makes the module being reviewed so large and/or complex that analysis of the code is difficult or impossible.
3) Adds functionality to the module, so that the module significantly extends the module's original functional description or specification.

Any conditional code that does **any** of these must be flagged as a discrepancy against the appropriate requirements cited above.

For the rest of the conditional code that will not be included in the production build, the reviewer must ensure the method used in the Trusted Build to keep the conditional code from compiling (IDE options, compiler flags, make files, etc) actually works as intended (that is, isn't overridden by another option, flag or code). Any issues or concerns found must be raised as a discrepancy or communicated to the people doing the trusted build to be sure the Trusted Build is done correctly.

Because the above recommendation will greatly increase the amount of time spent by the ITA to properly test source code that contains large amounts of conditionally compiled code, NYSTEC recommends that vendors strive to submit code that limits conditionally compiled code. NYSTEC views this as a best practice for voting systems and will result in decreased testing cost for voting system vendors. Again, we cannot require this but we do strongly recommend it.