# As-Built Software Requirements

**Objectives**: Prepare the OpenETran program for release under an open-source license, archived in a public source code repository. Update OpenETran and interface it with IEEE Flash, so that other developers may contribute and anyone may use the program.

1. The OpenETran software shall run on Windows XP or later.
2. The OpenETran software shall not use any commercial third-party components.
3. Microsoft Visual Studio 2010 shall be used for development.
4. The open source license type shall be GPL version 3.
5. The open source GNU Scientific Library (GSL) version 1.15 shall be used for linear matrix solutions, eigensystem solutions, and critical current iterations.
6. The software shall write waveforms in comma-delimited text, tab-delimited text, and the existing binary ELT formats.
7. The software shall support an iterative mode to find critical current for strokes to exposed poles and conductors.
8. The software shall support a text-based console execution mode, for both plotting and critical current iterations.
9. The software shall support execution from Microsoft Excel Visual Basic for Applications, version 2007 or later.
10. A template Excel VBA file shall be provided for IEEE Flash to produce OpenETran models from user inputs on pole/tower, surge arrester, grounding, conductor, span, and environment data in Excel.
11. The Excel VBA file shall critical current, arrester duty, phases flashing over, plot data and other numerical outputs from OpenETran.
12. The OpenETran software shall be tested and verified to produce matching outputs in console mode, for 27 existing test cases from the EPRI LPDW project.
13. The OpenETran / IEEE Flash package shall be tested on 3 cases:
    1. 15-kV wood crossarm line, from IEEE Std. 1410
    2. 35-kV wood pole structure with overhead shield wire, from IEEE Std. 1410
    3. 13.8-kV line with line arresters, from Chapter 14 of “Insulation Coordination for Power Systems” by A. R. Hileman, which contains both analytical results and ATP simulation results.
14. A separate installer shall be provided for a standalone version of OpenETran.
15. The software documentation shall include:
    1. Updated software requirements
    2. Build instructions and supporting narrative
    3. Change log, which is derived from Subversion file check-in comments
    4. Updated license file, release notes, and OpenETran user manual as needed
    5. Test case document, including instructions to run the test cases and expected results

# Build Instructions

The SVN repository is located at

<https://openetran.svn.sourceforge.net/svnroot/openetran>

As a preliminary step, the developer should also download and build a VC2010 version of the GNU Scientific Library (GSL). The OpenETran solution workspace assumes this was done at the default location of c:\gsl\gsl-1.15\

The solution workspace is called OpenETran.sln, and it builds both Debug and Release versions in 32 bits. All of the component (e.g. arrester, insulator) model code is located under the Components filter in the workspace. The higher-level source code files manage the overall solution process and outputs.

In order to update the software distribution, update these key files in the SVN root:

* OpenETran.exe – the Release mode executable
* OpenETran.pdf – User manual created from Doc\OpenETran.docx
* Readme.txt – release notes
* License.txt – probably will not change
* OpenETran.xlsm – the spreadsheet interface with VBA code.
* Input and script files under the test subdirectory

Then, freshen the OpenETran.zip archive with updated files, taking care to add any new ones from the test subdirectory or other locations. Copy the archive to SourceForge for release, and also to the Distrib folder for archiving in SVN.

# Test Notes

The three test cases referenced in Software Requirements were embedded into the spreadsheet interface as user examples.

The test\elt subdirectory archives binary plot and text file outputs derived from the last EPRI-released version of the standalone transients program, from LPDW v5 and SDWorkstation in 2001. That old version of the transients program is also archived in SVN, as elt.exe. In order to compare results, copy \*.dat files from the test directory into the test\elt subdirectory, and then execute runoldtests.bat from test\elt to run 27 test cases. Observe:

* The older version runs noticeably faster
* The initial parts of the output files do not match, because the Numerical Recipes library provided eigenvalues in a non-defined order, while GSL provides them in ascending or descending order. Therefore, the modal transformation matrices don’t match for a text comparison, but a close inspection shows that each of them will match after row and column transpostions.
* The key engineering outputs, such as insulator SI , peak voltages, and arrester energy all match.

Therefore, the converted program gives the same answers as the original program.

# Developer Notes

The user manual and its references document the model assumptions and solution techniques.

OpenETran could be expanded to cover more transient analysis applications, like TRV and capacitor switching. There could be a need for such features in an open-source program. (Note: while ATP is free of charge to those who can be licensed, it is not open-source, which puts some uncertainty on the continued support and availability of ATP). At present, OpenETran already offers:

1. Unbalanced Bergeron line models (like KC Lee in ATP)
2. RLC components
3. Non-linear arresters, insulators, and grounds
4. Sources
5. A SPICE-like interface for developing component code models
6. Ability to increase the time step during arrester duty simulations
7. Source code that should be readily portable to Linux

A Stage 1 expansion might include:

1. AC initial conditions
2. Switches and circuit breaker models
3. Transformer models

A follow-on Stage 2 expansion might include:

1. Frequency-dependent phase-domain line models, such as Gustavsen’s
2. Interface to OpenModelica or VHDL-AMS, allowing custom component and control models
3. Rotating machine models built in

Some architectural changes would also be desirable:

1. Change the input syntax for compatibility with OpenDSS
2. Document the procedures for adding modular code models
3. Implement a variable time step