Developing STP for Windows and XML

Introduction

During my eleven-week internship as a SURE intern at the USGS, I was given a few projects to work on; to develop a console version of STP for Windows, a GUI version of STP for Windows, and to add a feature to STP allowing for users to attain certain data in XML format.

STP 1.4 for Windows

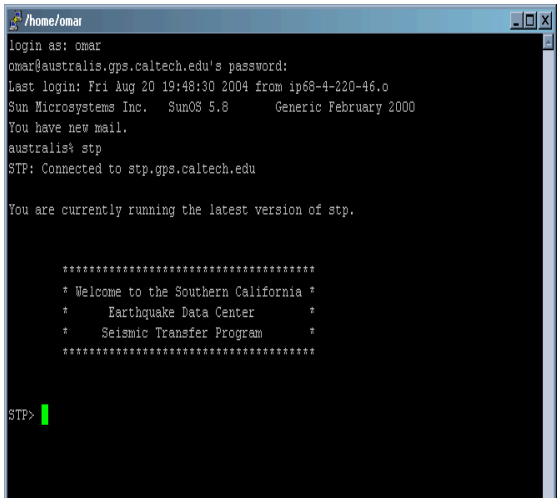
Because there was a not version of the Seismogram Transfer Program (STP) for Windows, I was instructed to develop just that. I created two versions of STP; a console program that worked identical to the UNIX/Linux version of STP, and a GUI version of STP, that was loosely based on the Internet based version of STP. Coding went smooth for the Windows console version of STP. After looking over the code for the UNIX/Linux version of STP, I figured out that I only needed to change a few functions and add a few headers to make it work on the Windows environment. In the UNIX/Linux version, data was being sent to and received from the server via file transfers. However, that would not work on the Windows environment. Therefore, it was necessary for me to use raw socket functions to send and receive data from the server.

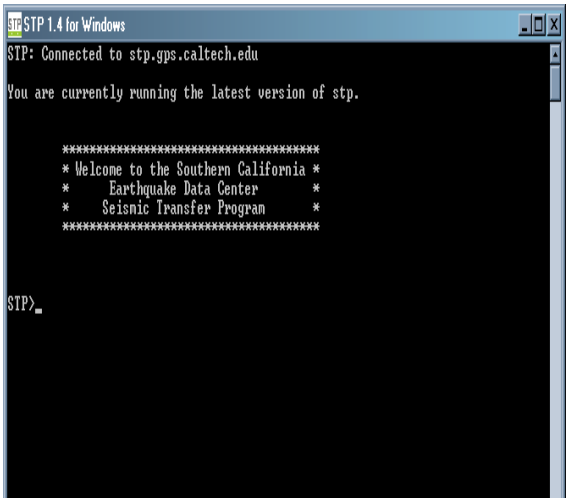
When I was done with the console program, it was now time to develop a GUI version of STP. As a reference, I looked at the web-based version of STP that is coded in Java and decided to model my GUI layout after that. Using the

Windows API, I was able to create the GUI for my application. Having never coded a Win32 application, it seemed confusing to me at first. However, after a lot of reading through the Internet, books, and the MSDN website, coding using the Win32 API did not appear to be too difficult.

A few problems did arise when during the coding of the GUI version of STP. Since I already had all of the client/server functions coded in the console program, I added that into the GUI version, and that caused a problem. Whenever a command was sent to the server, the program would stop responding until it was done receiving data from the server. Obviously, something needed to be done about this, so I used multithreading to fix that problem. However, once I did that, other functions began to work in such a way that I did not want them to, and I had to change how they functioned. Nonetheless, after some debugging, I was able to fix those problems.

Below are Figures 1 and 2, which show how the UNIX/Linux version of STP is comparable to the Windows version, and how the internet version is comparable to the Windows GUI version.





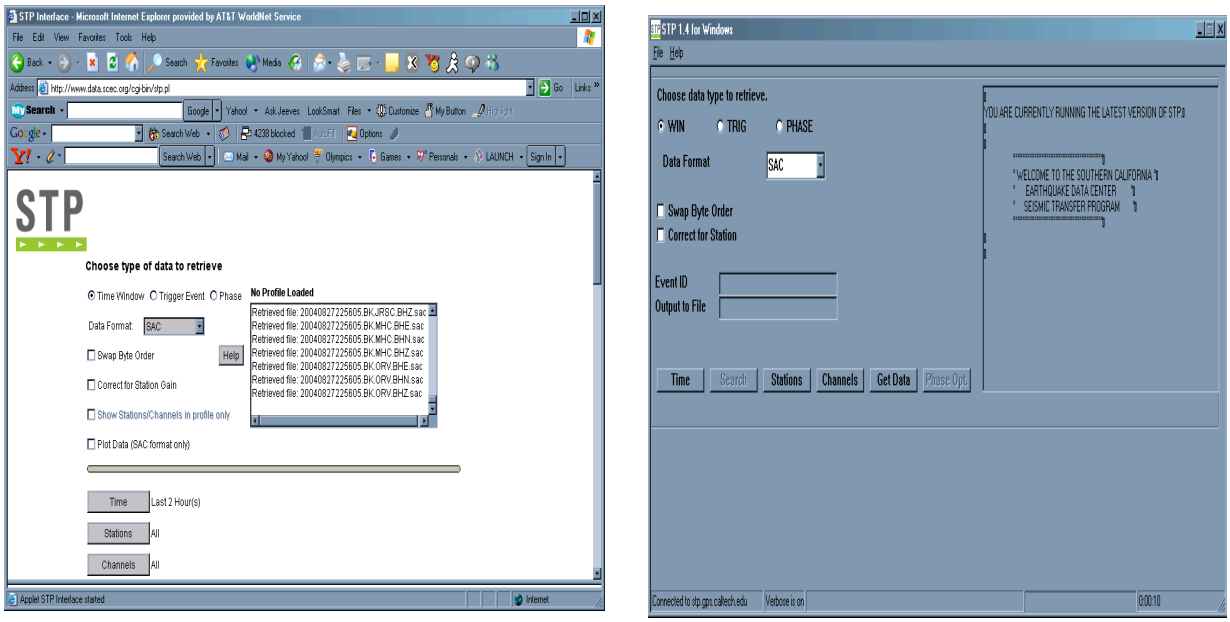


Figure 2: Interfaces of the web and Windows client GUI versions of STP

How to Run STP for Windows

To attain a copy and run the console version of STP for Windows, perform the following steps:

1. go to http://www.data.scec.org/ftp/programs/stp
2. left click on stp.exe
3. click on save file to disk
4. double click on stp.exe

You can enter in commands just as you would the UNIX/Linux version of STP. Any data downloaded will be saved in the same directory you are running stp.exe from. As an example, at the STP prompt, enter in the command:

STP> PHASE –f northridge.txt –e 3144585.

This command will save a file (the -f command) called northridge.txt containing phase information for the event (the -e command) 3144585, the event ID for the 6.7 Northridge event to your working directory.

To attain a copy and run the GUI version of STP for Windows, perform the following procedures:

1. go to http://www.data.scec.org/ftp/programs/stp.
2. left click on stp_gui.exe.
3. click on save file to disk
4. double click on stp.exe

The GUI version will allow the user to select from entering in a Win, Trig, or Phase command. As well as allowing the user to search for certain events while in Trig or Phase modes. Again, any downloaded will be saved in the same directory you are running stp_gui.exe.

Adding XML to STP

After creating the console and GUI versions of STP for Windows, I was assigned to add an option to allow for users to receive Event and Phase data from the server in XML format. I first created an XML schema for the Phase and Event commands, which is used to validate the XML document. I then coded the parser for the Event and Phase commands so that when the user entered in a Phase or Event command, the data would be stored in XML format.

The do_event.c file, which handled the Event commands, for the most part, already had the XML Parser coded in it. I made a few minor changes to the parser so that it would be valid against my Event schema. Since the Event parser already coded, coding the Phase parser went smoothly. After coding the parser it was time to see if the code would work and if it would be valid against

my schemas. To my delight, the XML files were indeed correct. Now it was time to document my work and put it up on the SCEDC web page.

I created an Introductory page about STP and XML, with links to the documentations and schemas for both the Event and Phase commands. Figure 3 below shows my schemas for the Phase and Event commands. For more information on my project of adding XML to STP, go to http://www.data.scec.org/xml.

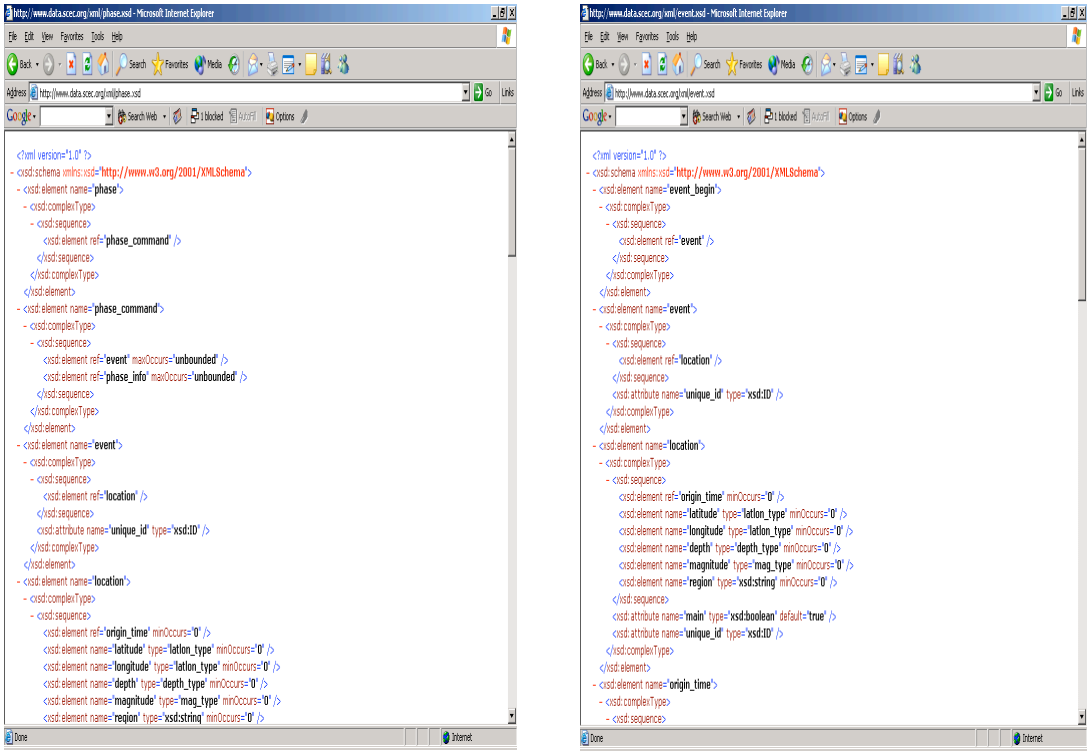


Figure 3: XML Phase and Event Schemas

How to Receive Data in XML Format

To receive data in XML format you must enter in xml at the stp prompt. For example:

STP> XML

Now you can enter in a Phase or Event command as you normally would, and the XML files should be saved to your working directory.