ReconServer User Manual

Mark Cianciosa
Auburn University Department of Physics

April 19, 2012

Contents

List of Tables

List of Figures

1 Introduction

ReconServer is a program designed to abstract the details of running reconstructions from a user. This abstraction is made possible by a messaging protocol implemented in the standard Transmission Control Protocol(TCP) that allows operation of reconstructions from any user anywhere in the world. Through the use of TCP a client program can be created from any device or language that has TCP support.

2 Server Structure

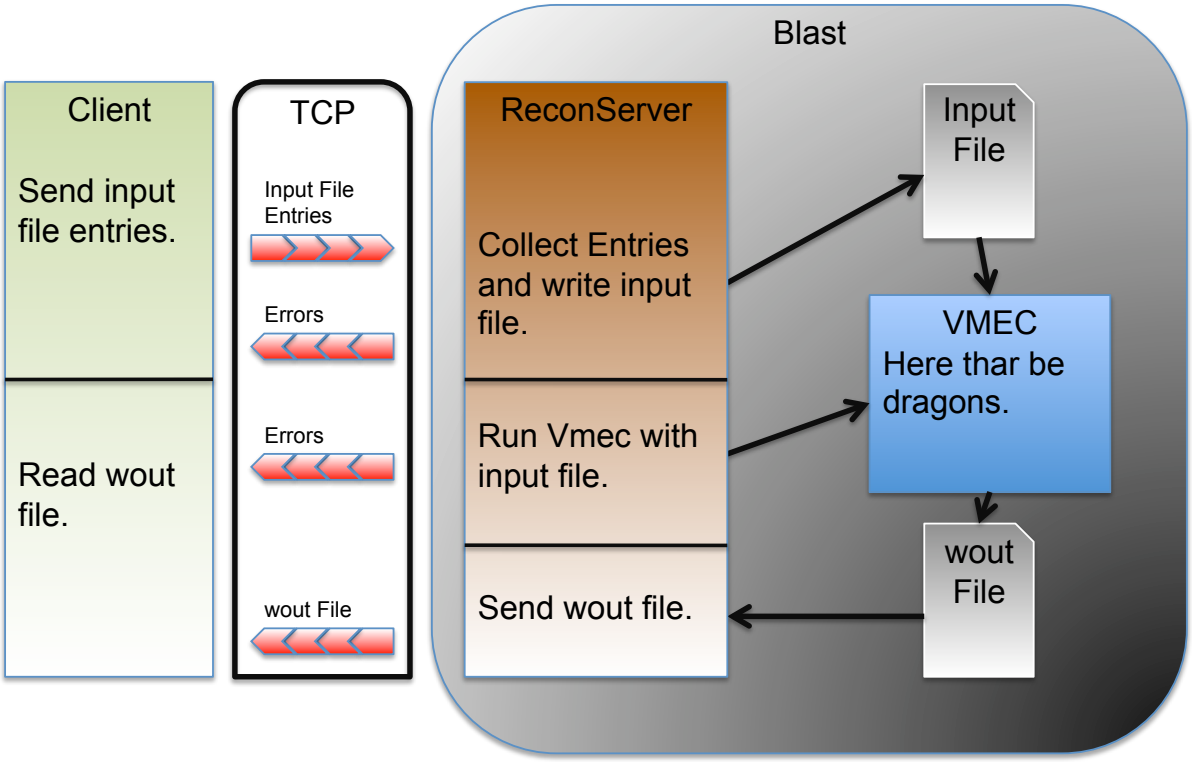

Figure 1: Map Of ReconServer

The ReconServer (figure 1) acts as a coordinator providing a single point between a connecting client and the reconstruction code. The server is a continuously running program what waits for a client to connecting using TCP sockets on port 2000. When a client connects the server spawns a thread that takes over all further communication with the client. The main thread then resumes waiting for the next client to connect. This allows for multiple clients to connect to the server simultaneously. A client may also create multiple connections from the same client program. In order to avoid conflicts each thread is assigned a unique identification number to insure that each input file and output file remains associated with a particular client.

Once connected, a client sends a list of input file entries. The server creates a series of input files based on when current date, shot number, and time slice When finished the server

creates an input file. If the server has successfully written a file with no errors the server sends a no error message to the client. If in the event an error has occurred the server sends an error message. If an error has occurred if should be assumed by the client that this connection is no longer valid and disconnect. When an error occurs in a spawned thread it only affects that particular thread. The main server thread and other spawned threads will not be affected.

With a completed input file the server runs the reconstruction and waits until computation has finished before continuing. At the successful completion of the reconstruction the server reports back no errors if none have occurred. The server then sends the client the computed `wout` file. The server will again report if any errors occurred then sent the `threed1` file then disconnect from the client. The server will delete any intermediate files that have been written in the course of computation unless this function has been disabled.

3 Administrator Setup

3.1 System Requirements

ReconServer is written in java using Java SE 1.5 runtime or greater. The java system is available from `http://www.java.com`.

3.2 Compiling

To compile ReconServer requires Java SE 1.5 or better and `ant` available from `http://ant.apache.org/`. In the main directory there is a `build.xml` file. ReconServer can be compiled using the command

```
ant
```

The resulting compiled server can be found in the `dist` directory. Modifications including new source files should be added to the `src` directory. Any changes to the source require no modifications to the `build.xml` file. To clean up intermediate files from compiling use the command

```
ant clean
```

Because of the multi-platform nature from java the source can be compiled on any platform and the resulting ReconServer.jar file can be executed on any platform.

3.3 Installation

To install ReconServer copy `ReconServer.jar` to any directory. Set the `PATH` environment variable to the directory that `ReconServer.jar` is installed in. On Solaris set the `.jar` file to be executable using the command.

```
chmod +x ReconServer.jar
```

3.4 Running

To start the server type the command

```
java -jar <path to>ReconServer.jar <-flags>
```

If `ReconServer.jar` is pointed to by the `PATH` environment variable it is unnecessary to give the path. Alternatively if `ReconServer.jar` has been set to be executable on Solaris or if it is being run on Windows, the server maybe started simply by typing

```
ReconServer.jar <-flags>
```

If the server has been created from the command line the server can be stopped by typing

```
<control> c
```

On Unix the server can also be stopped by typing the command

```
ps -ef | grep java
```

to find the process id number. Then type

```
kill <idnumber>
```

to end that process.

3.5 Command Line Flags

`-h` Displays all the supported command line flags.

`-p <port>` Sets the port number for the server. The default is `2000`.

`-c <command>` Sets the command to run the reconstruction. The default is `xvmec2000`.

`-o` Turns on verbose output. This displays diagnostic information while the server is running.

`-d` Enables the deletion of files and directories left over from the reconstruction.

`-f <prefix> <postfix>` Specifies which output files to send back to the client. Server uses the prefix to search for the file. The postfix is arbitrary and will be tagged onto end of the file. The postfix my be empty by setting the argument to `""`.

4 Directory Structure

In order to separate different reconstructions the server setup a hierarchy of directories to store input and output files in. Starting with the current date `YEAR:MONTH:DAY:`. The server will create nested directories for shot number and time slices `SHOT NUMBER:TIME SLICE:`. With in the time slice directories starting from zero are created for each attempt of that specific reconstruction of that shot number and time slice on a given day. If the deletion of directories has been enabled a directory is only deleted is that directory is empty. The directories are (table 1)

`YEAR`	All reconstructions with in a given year.
`MONTH`	All reconstructions with in a given year.
`DAY`	All reconstructions on a given day.
`SHOT NUMBER`	All reconstructions of a shot number performed on a specific day.
`TIME SLICE`	All reconstructions of a specific time slice of a shot number.
`ITERATION`	Directories starting from 0 separating different reconstructions.

Table 1: Directory Structure

5 Messaging Protocol

5.1 Input File Entries

The input protocol consist of only input file entries. All input entries begin with a name of the entry. The name starts with the number of characters in the string represented as a 32bit integer. Then the string as an array of 8bit integers or chars.

```
size<int32>char[1]<int8>char[2]<int8>.....char[size]<int8>
```

All input files must start with two entries. The first entry must be named `Shot Number` followed by the second entry `Time Slice`. The data for these entries can be of any type. If these two entries are not the first two entries the server informs the client there is an error and the input file is not written. The input file is completed with an entry name `EOF`. Once the EOF entry is reached the server ignores all further input from this connection. As such there this no need to send anything beyond the name for the `EOF` entry.

The data entries must be one of 14 types. The available types are (table 2). All data messages start with an 8bit integer to set the data type. For scalar types it is immediately followed by the data.

```
datatype<int8>data<data type>
```

For arrays and strings the data type descriptor is followed by the size, as a 32bit integer, of the array or string then the data as each element of the array or string.

Type Name	Data Size	Type Description Value
Comment	n/a	-1
Boolean	8bits	0
Int8	8bits	1
Int16	16bits	2
Int32	32bits	3
Int64	64bits	4
Float	32bits	5
Double	64bits	6
1D Boolean Array	8bits*(Size of Array)	7
1D Int8 Array	8bits*(Size of Array)	8
1D Int16 Array	16bits*(Size of Array)	9
1D Int32 Array	32bits*(Size of Array)	10
1D Int64 Array	64bits*(Size of Array)	11
1D Float Array	32bits*(Size of Array)	12
1D Double Array	64bits*(Size of Array)	13
String	8bits*(Size of String)	14
1D String	8bits*(Size of String) per Element	15

Table 2: Data Value Types

```
datatype<int8>size<int32>data[1]<type>data[2]<type>......data[size]<type>
```

For arrays of strings, the data type descriptor is followed by the size of array as a 32 bit integer. Each element of the array is read as a 32bit integer for a size of the string followed by the string data it self.

```
datatype<int8>size<int32>size[1]<int32>data[1]<string>
...size[size]<int32>data[size]<string>
```

5.2 Errors

Through out the course of computation errors can arise. In the event of an error ReconServer has an implemented error messaging protocol. If no error has occurred a `NO ERROR` message is sent in its place. To insure that data communications are synchronized between the client and server the client must check for errors. Errors must be checked twice after the input file entries are sent. The first error message reports on the state of input file. The second message reports on the state of the reconstruction's computation. Error messages will also be sent after the `wout` and `threed1` files are sent. If no error has occurred while sending the `threed1` file, the server will disconnect immediately. It is not necessary to check for errors after the `threed1` has been sent. Error messages are started by sending a 32bit integer to specify the error number. This is followed by a second 32bit integer representing the size as number of characters in the

error message and lastly the error message as a string of 8bit integers.

`errorcode<int32>size<int32>char[1]<int8>char[2]<int8>.....char[size]<int8>`

Possible errors are (table 3). Between these two error messages client needs to stall while

Error Type	Error Code	Message
No Error	0	
Generic Error	1	An error occurred in Recon Server
Input File Start Error	2	Bad start of input file. All reconstruction input files must start with a `File Type`, `Shot Number` and `Time Slice` entries.
Close File Error	3	A failure occurred when trying to close the reconstruction input file.
Entry Name Error	4	A failure occurred when trying to read a reconstruction input file entry name. A possible cause is that the end of the file was reached with out an EOF command sent.
Entry Value Error	5	A failure occurred when trying to read a reconstruction input file entry value.
Create File Error	6	A failure occurred when trying to create the reconstruction input file.
Write File Error	7	A failure occurred when trying to write the reconstruction input file.
Execution Error	8	A failure occurred when trying to run the reconstruction.
Outfile File Error	9	A failure occurred when trying to read an output file. A possible reason is the file was never created.
Invalid Input type Error	10	A failure occurred when trying an input entry of an unsupported type. Supported types are shown in table 2.
Invalid Input file Error	12	A failure occurred when trying to create the input files. The server only supports vmec and v3fit inputs.

Table 3: Errors

Vmec finishes computation.

5.3 Output Files

To read back the `wout` and `threed1` files the message starts by sending the file name. First a 32bit Integer is sent representing the size of the filename. The file name is then sent as a string of 8bit Integers. The file name maybe changed without affecting the `wout` or `threed1` file data.

`size<int32>char[1]<int8>char[2]<int8>.....char[size]<int8>`

The actual file it self is then sent as a 64bit integer representing the number of bytes in the file followed by an array of 8bit integers. In order to reconstruct the file correctly is must be written to disk as a series of 8bit integers in the exact order.

`filesize<int64>data[1]<type>data[2]<type>.....data[size]<type>`

After the file `wout` is sent the `threed1` file is sent in the same manor. The `threed1` maybe ignored by the client with no disruption of server operation. Once all files have been sent no further communication with server is necessary or possible.

All data must be sent and received as big endian or network format. If a client is being implemented on a little endian platform it may be necessary to preform a byte swap before messages can be sent or after they are received. This step may not be necessary if functions for accessing TCP implement this functionality automatically.

6 Labview Client Programing

A set of Labview vi's have been created to implement the messaging protocol.

6.1 Reconstruction Open Connection

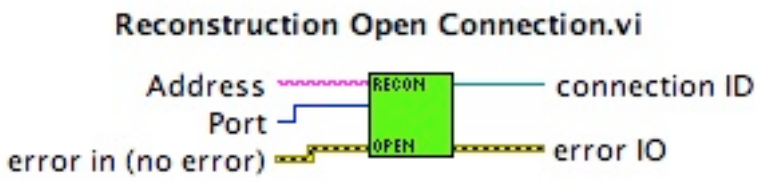

Figure 2: Reconstruction Open Connection

Creates a TCP connection to the server specified by `Address` on the port number specified by `Port`. The default values are `blast.physics.auburn.edu` on port `2000`. Multiple connections can be created in parallel in a single VI. Each instance is given a unique `connection ID` to track each connection. (figure 2)

6.2 Reconstruction Create Input File

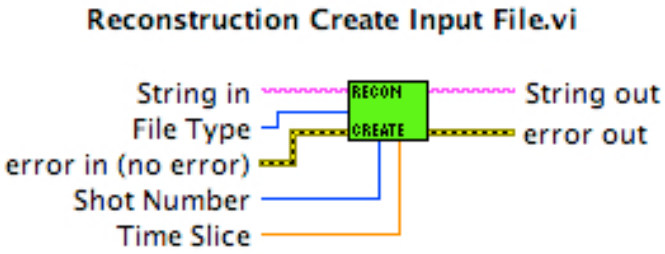

Figure 3: Reconstruction Create Input File

Sets consecutive input entries for `File Type`, `Shot Number` and `Time Slice`. These three entries must be at the start of every input file however they are not valid reconstruction inputs. When using multiple input files, this is used to mark the start of new input file. Only two types of input files are supported `vmec` and `v3fit` inputs. `Shot Number` and `Time Slice` are used to define the filename. All input file entry lists must start with `File Type`, `Shot Number` and `Time Slice` command or an error will be returned. (figure 3)

6.3 Reconstruction Write Execution Parameters

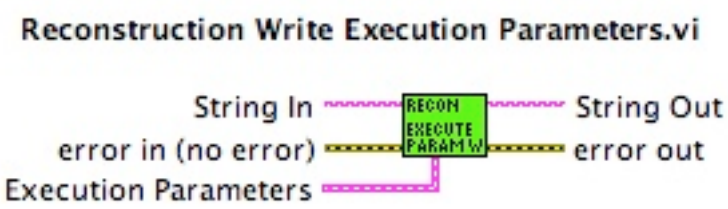

Figure 4: Reconstruction Write Execution Parameters

Sets various entries related to execution of the reconstruction program. (figure 4) (table 4)

Name	Default Value	Data Type
LFORBAL	F	Boolean
LFREEB	T	Boolean
DELT	0.70	Double
TCON0	2.0	Double
NFP	5	Int32
NS_ARRAY	8, 15	1D Int32
FTOL_ARRAY	5.0E-8, 5.0E-14	1D Double
NITER	2500	Int32
NSTEP	200	Int32
NTOR	4	Int32
MPOL	5	Int32
NZETA	32	Int32
NVACSKIP	9	Int32
MGRID_FILE	mgrid_cth.dr13s.nc	String
LASYM	F	Boolean

Table 4: Execution Settings

6.4 Reconstruction Write Coil Currents

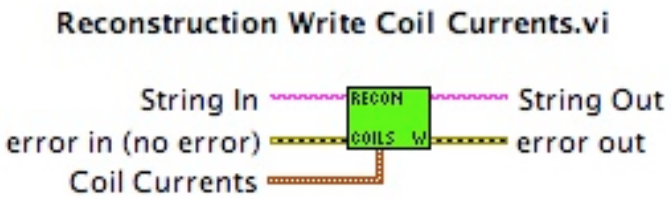

Figure 5: Reconstruction Write Coil Currents

Sets the CTH coil currents. By default all currents are zero. (figure 5)

6.5 Reconstruction Write Initial Positioning

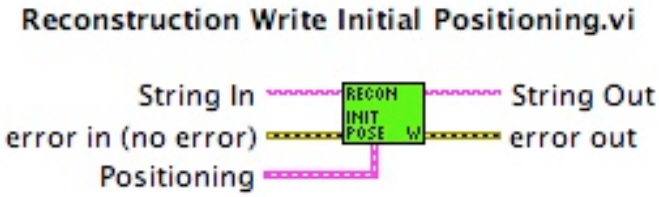

Figure 6: Reconstruction Write Initial Positioning

Sets various entries related to initial position. (figure 6) (table 5)

Name	Default Value	Data Type
`RAXIS(0)`	`0.7275`	Double
`ZAXIS(0)`	`0`	Double
`RBC`	`0.698, 0.131`	1D Double
`ZBS`	`0, 0.2625`	1D Double

Table 5: Positioning Settings

6.6 Reconstruction Write Fit Parameters

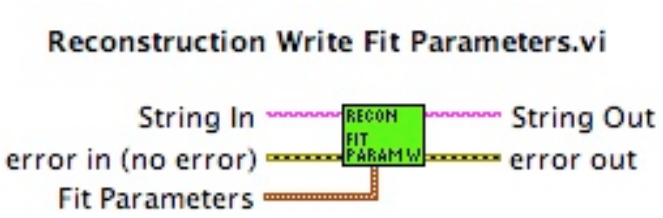

Figure 7: Reconstruction Write Fit Parameters

Sets various entries related to model fitting. (figure 7) (table 6)

Name	Default Value	Data Type
GAMMA	0.0	Double
PHIEDGE	-0.03	Double
BLOAT	1	Double

Table 6: Fit Settings

6.7 Reconstruction Write Plasma Currents

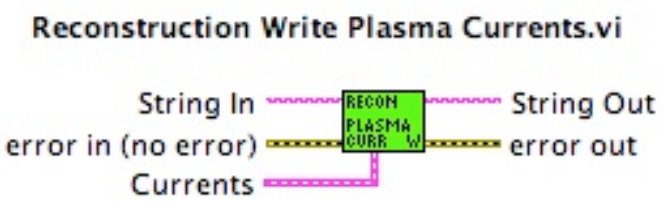

Figure 8: Reconstruction Write Plasma Currents

Sets various entries related to the plasma currents. (figure 8) (table 7)

Name	Default Value	Data Type
NCURR	1	Int32
CURTOR	0.0	Double
AC	1.0, 4.0, -5.0, 0.0, ...	1D Double

Table 7: Plasma Currents Settings

6.8 Reconstruction Write Plasma Pressure

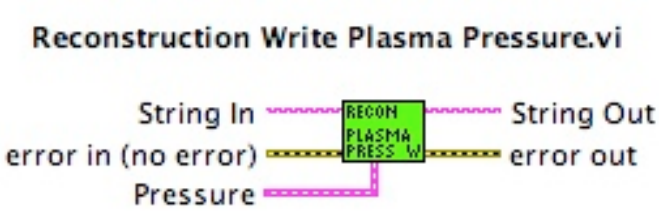

Figure 9: Reconstruction Write Plasma Pressure

Sets various entries related to the plasma pressure (figure 9) (table 8)

Name	Default Value	Data Type
SPRES_PED	1	Double
AC	5.0E-5, -1.0E-4, 5.0E-5	1D Double

Table 8: Plasma Pressure Settings

6.9 Reconstruction Set

Convenience function for setting individual inputs can be found in the sub-pallet underneath each reconstruction input vi as `Reconstruction Set <input>`.

6.10 Reconstruction Write

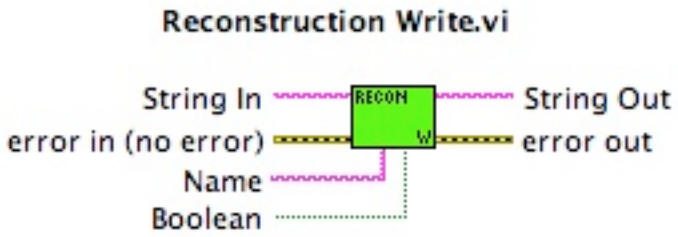

Figure 10: Reconstruction Write

Adds an input file entry to string of input file entries. This is a polymorphic vi that supports all 14 input data types (table 2) that ReconServer implements. Note that these functions cannot be used to write an input entry that has been written by one of the `Reconstruction Write Input` vis. If a client needs to set a few inputs manually found in such a vi then client must implement all the inputs that are written from that vi. (figure 10)

6.11 Reconstruction Write EOF

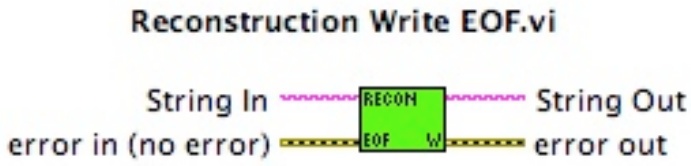

Figure 11: Reconstruction Write EOF

Adds an `EOF` input file entry to the input file entries string. Once the Server detects an `EOF` the server ignores all further input from the client. All input file entry lists must end with an `EOF` command or an error will be returned. (figure 11)

6.12 Reconstruction Write Command

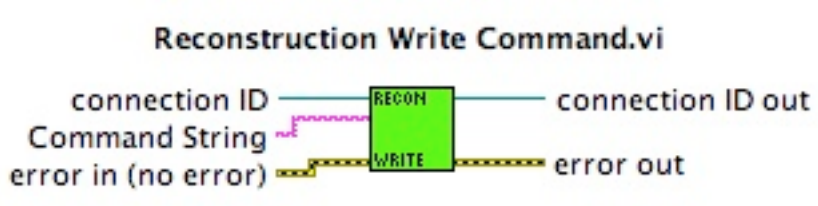

Figure 12: Reconstruction Write Command

Sends the input file entry list to the server specified by connection ID. It is up to the user to create a valid reconstruction input file. ReconServer then reads the entries and returns any errors. The vi reads back any errors and disconnects if an error should occur. (figure 12)

6.13 Reconstruction Read Output File

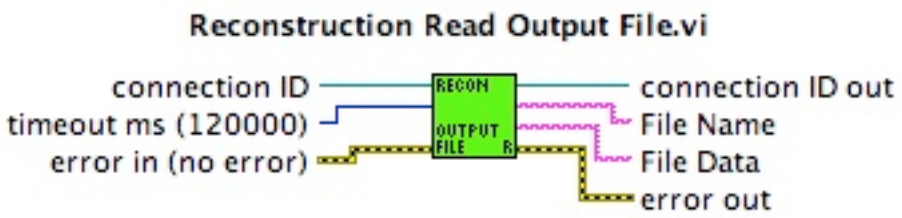

Figure 13: Reconstruction Read Output File

This vi will stall communication of its connection until Vmec has finished computation. If any errors should occur or if Vmec has failed to run an error will be read back and the client will disconnect. If no errors occur the output file will be read back as a string of 8bit integers with the accompanying file name. This can be called twice. The first call will return the `wout` file. The second call will return the `threed1` file. (figure 13)

6.14 Reconstruction Close Connection

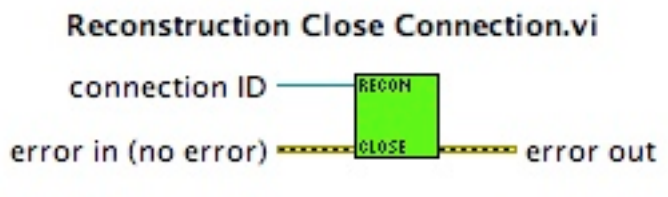

Figure 14: Reconstruction Close Connection

Closes the connection specified by connection ID. (figure 14)

6.15 Reconstruction Write Output File To Disk

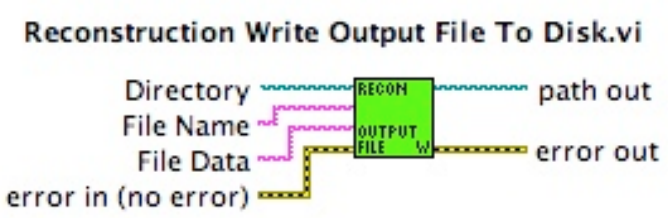

Figure 15: Reconstruction Write Output File To Disk

Writes the output file from `Recon Read Output File` to a local disk specified by Directory. The file data must not be altered in any way or else the file may not be read back properly. The file name may be changed to anything. The returned path includes the file name. (figure 15)

7 Known Issues

7.1 Input Files

The server cannot check if the input file is a valid input up file. If an improper input file is sent Vmec will not generate a wout File.

7.2 Output Files

If Vmec fails to create an output file there is no way to report back the error back to the client. The server will disconnect from the client or the client will read nothing instead. In the Labview library a disconnect error is detected from `Recon Read Output File`.

7.3 Port In Use

Some times when ReconServer has not shutdown properly the Server does not relinquish control of port `2000`. Subsequent running of ReconServer will report that port `2000` is in use. In the event this happens the ReconServer can be shut down in the following way. Type the command

```
ps -ef | grep java
```

to find the process id number. Then type

```
kill <idnumber>
```

to end that process.

7.4 Write Input File Entries

The input file entries do not get over written. A client must choose between using the built in function to get input file entries or create them manually.

7.5 Off Campus Connections

Clients can connect to the server on blast however if the client doesn't have an Auburn University ip address the client will not communicate with the server. To access off campus run the CISCO VPNclient and log into Auburn's off campus connection.

7.6 Unable To Access Jarfile

When executing the jar file from a different directory than the directory from which the jar file is stored using the command,

```
java -jar ReconServer.jar
```

the path to the jar file must be specified if the ReconServer.jar is not pointed to in the `PATH` environment variable.