U.S. DEPARTMENT OF ENERGY

PNNL-22605

Prepared for the U.S. Department of Energy
under Contract DE-AC05-76RL01830

The ReadStream Utility Class

DES-0003

Revision 1

Charlie Hubbard

March 2010

Pacific Northwest
NATIONAL LABORATORY

Proudly Operated by **Battelle** *Since 1965*

The ReadStream Utility Class

Charlie Hubbard

DES-0003
Revision 1
March 2010

DISCLAIMER

PACIFIC NORTHWEST NATIONAL LABORATORY
operated by
BATTELLE
for the
UNITED STATES DEPARTMENT OF ENERGY
under
Contract DE-AC05-76RL01830

Printed in the United States of America

Available to DOE and DOE contractors from the Office of Scientific and Technical Information, P.O. Box 62, Oak Ridge, TN 37831-0062
ph: (865) 576-8401
fax: (865) 576-5728
email: reports@adonis.osti.gov

Available to the public from the National Technical Information Service, U.S. Department of Commerce, 5285 Port Royal Rd., Springfield, VA 22161
ph: (800) 553-6847
fax: (703) 605-6900
email: orders@ntis.fedworld.gov
online ordering: http://www.ntis.gov/ordering.htm

Contents

1 Introduction

This document describes the `ReadStream` utility class which is defined and implemented in *readStream.h* and *readStream.cpp* respectively.

The system control software is implemented as a series of cooperating client and server applications that all run concurrently on the control computer. The applications communicate with each other by passing text-based messages back and forth to one another over TCP connections. Messages are encoded using our data serialization protocol (DSP) (see design document ***DES-0002*, *Data Serialization Protocol*** for complete details).

DSP format messages are variable in length and contain no length field in them to allow the receiver to know beforehand how many bytes need to be read off the wire before the complete DSP message has been received. Instead, receivers are expected to count opening and closing parentheses, declaring a message to be complete once the closing parenthesis corresponding to the message's initial opening parenthesis is encountered. This is a somewhat cumbersome task that needs to be carried out in a number of different places in the code base. The `ReadStream` class exists to make this message recognition task easy and consistent across the entire code base.

2 The `ReadStream` Class

The `ReadStream` class definition is shown below.

```
class ReadStream {
public:
   ReadStream();
   bool Read(const int sockId, string &msg);

   string lastErrorMessage;
};
```

As you can see, it has about as simple an interface as a class could have, providing just one method — `Read()`.

2.1 Usage

Users of the class are expected to already have a TCP connection established on which DSP format messages are expected to be received. To receive the next available DSP message from the connection, they simply call the class' `Read()` method, passing the socket descriptor of the TCP connection as the first argument, and providing an STL string to hold the received message as the second argument. If the call is successful (a valid DSP message was received), then the method returns `true`. Otherwise `false` is returned. If the call fails (i.e. if `false` is

returned), the class' attribute `lastErrorMessage` will contain a human-readable description of the problem.

2.2 Implementation

Internally, the `Read()` method reads characters off the TCP connection one character at a time. It examines each character, looking for opening or closing parentheses. It maintains an internal counter called `nestLevel`, which is initially zero. Every time an opening parenthesis is encountered, this counter is incremented by one, and every time a closing parenthesis is encountered, the counter is decremented by one. Because all DSP format messages *must* begin with an opening parenthesis, the very first character received should increment the nesting level counter. Characters continue to be processed until the corresponding closing parenthesis is encountered, which is indicated by the nesting level counter being decremented to zero.

When the function begins, the user-provided message string is cleared. As characters are received off the wire, they are appended to the end of the message string so, by the time the nesting level counter decrements back to zero, the message string contains the text of the complete DSP message.

The opening and closing parenthesis characters can also appear inside a DSP message as a string literal rather than a grouping operator. In this case, the parenthesis is always preceded by a tilde character. The `Read()` method looks for tilde characters and properly recognizes parentheses that are literals (these have no effect on the nesting level counter).

Pacific Northwest
NATIONAL LABORATORY

Proudly Operated by **Battelle** *Since 1965*

U.S. DEPARTMENT OF
ENERGY

902 Battelle Boulevard
P.O. Box 999
Richland, WA 99352
1-888-375-PNNL (7665)
www.pnnl.gov