



## Department of Civil, Structural, and Environmental Engineering

### SEESL Structural Engineering and Earthquake Simulation Laboratory SEESL

212 Ketter Hall, North Campus, Buffalo, NY 14260-4300

Fax: (716) 645-3733 Tel: (716) 645 5400 X 16

<http://www.nees.buffalo.edu>

# SCRAMNet DAQ Developers Manual

by

Jason P. Hanley  
University at Buffalo

Report No. SEESL-2004-01

Version 1.0.0

University at Buffalo  
Buffalo, New York  
October 5, 2004

*Structural Engineering*

**SEESL**  
*Laboratory*

*Earthquake Simulation*

## Table of Contents

<b><u>1. INTRODUCTION.....</u></b>	<b><u>3</u></b>
<b><u>2. ABOUT SCRAMNET.....</u></b>	<b><u>3</u></b>
<b><u>3. DESIGN .....</u></b>	<b><u>4</u></b>
<b>3.1. SCRAMNET DAQ SERVER.....</b>	<b>5</b>
3.1.1. SCRAMNET INTERFACE.....	5
3.1.2. DATA LISTENER .....	6
3.1.3. CLIENT LISTENER .....	6
<b>3.2. SCRAMNET DAQ CLIENT .....</b>	<b>7</b>
3.2.1. UI .....	7
3.2.2. SERVER CONNECTION .....	7
<b><u>4. BUILDING .....</u></b>	<b><u>7</u></b>
<b>4.1. SCRAMNET DAQ SERVER.....</b>	<b>7</b>
<b>4.2. SCRAMNET DAQ CLIENT .....</b>	<b>7</b>
<b><u>5. INSTALLATION AND CONFIGURATION.....</u></b>	<b><u>8</u></b>
<b>5.1. SCRAMNET DAQ SERVER.....</b>	<b>8</b>
<b><u>6. RUNNING.....</u></b>	<b><u>8</u></b>
<b>6.1. SCRAMNET DAQ SERVER.....</b>	<b>8</b>
<b>6.2. SCRAMNET DAQ CLIENT .....</b>	<b>8</b>
<b><u>7. AUTHORS.....</u></b>	<b><u>8</u></b>
<b><u>8. ACKNOWLEDGEMENTS.....</u></b>	<b><u>9</u></b>
<b><u>9. LICENSE .....</u></b>	<b><u>9</u></b>

## Table of Figures

Figure 1: SEESL SCRAMNet Network .....	3
Figure 2: Software Components .....	4

## 1. Introduction

SCRAMNet DAQ is a system designed to read data off of the SCRAMNet shared memory network by Systran. It then can record and stream this data for archival and real-time viewing purposes.

This system is composed of two applications. One is SCRAMNet DAQ Server which actually does all the reading, recording, and streaming of data. The other is SCRAMNet DAQ Client which is the user interface to the server. The server runs on its own dedicated machine and the client connects to the server from any computer with a network connection.

## 2. About SCRAMNet

SCRAMNet is a network of computers that share a common memory. These computers are connected by fiber optic cables and use special hardware, developed by Systran, to map this shared memory into the user memory space of each computer system.

Below is the network diagram for the SEESL SCRAMNet network. We call this SCRAMNet 2 because we have another private SCRAMNet network dedicated to control of our equipment by our MTS controllers.

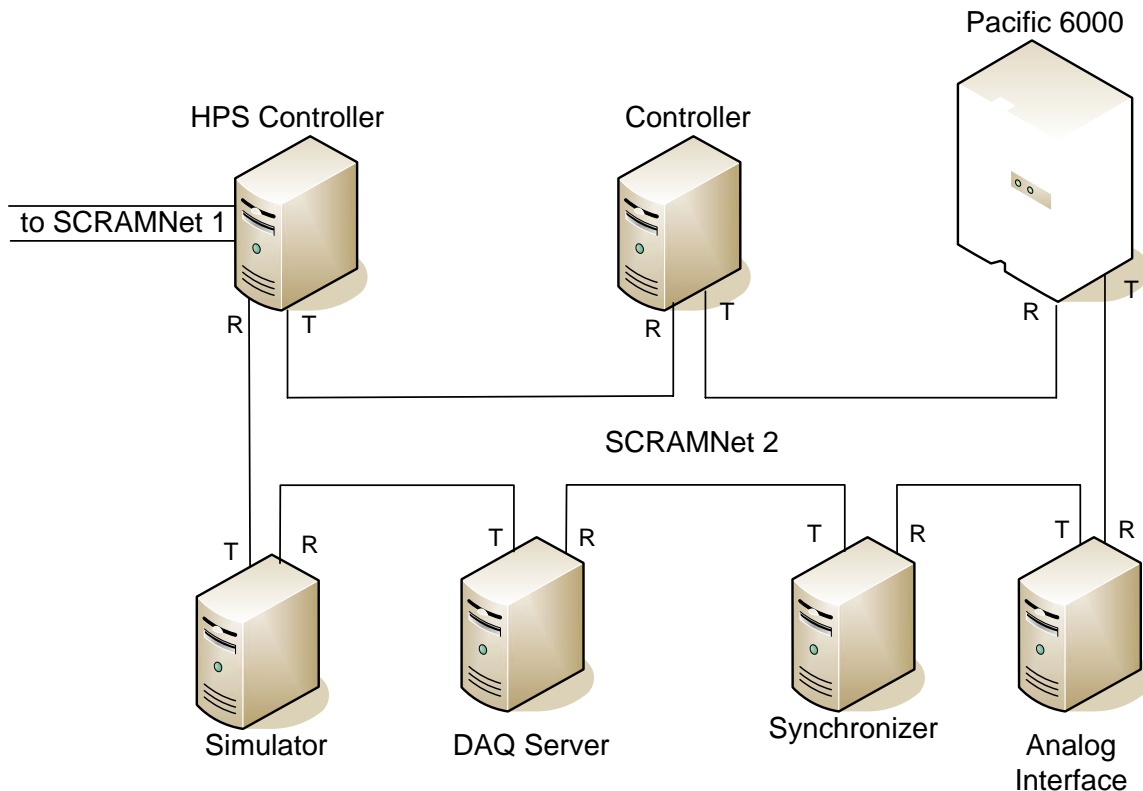


Figure 1: SEESL SCRAMNet Network

Each node in this network is assigned an ID but all the computers share the exact same memory when on the network.

A special feature of SCRAMNet that this system uses are network interrupts. These are interrupts generated on SCRAMNet when a certain event happens and then translated into a local interrupt on computers in the network that are interested in that specific interrupt. Interrupts are generated on SCRAMNet for things such as reads, writes, and errors.

Interrupts are a key part of this application when operating in synchronous mode. This mode allows us to derive a clock off of an interrupt from SCRAMNet that has been generated from some master. This clock can be used to determine when to take samples which are synchronized to this clock.

At SEESL, our MTS equipment generates a 1024 Hz clock that we use to capture data synchronously with our controllers.

### 3. Design

The system is broken down into two different applications: SCRAMNet DAQ Server and SCRAMNet DAQ Client. The server does all the work of sampling data, recording data, and streaming data. The client controls the server and provides a graphical user interface to the operator of the system.

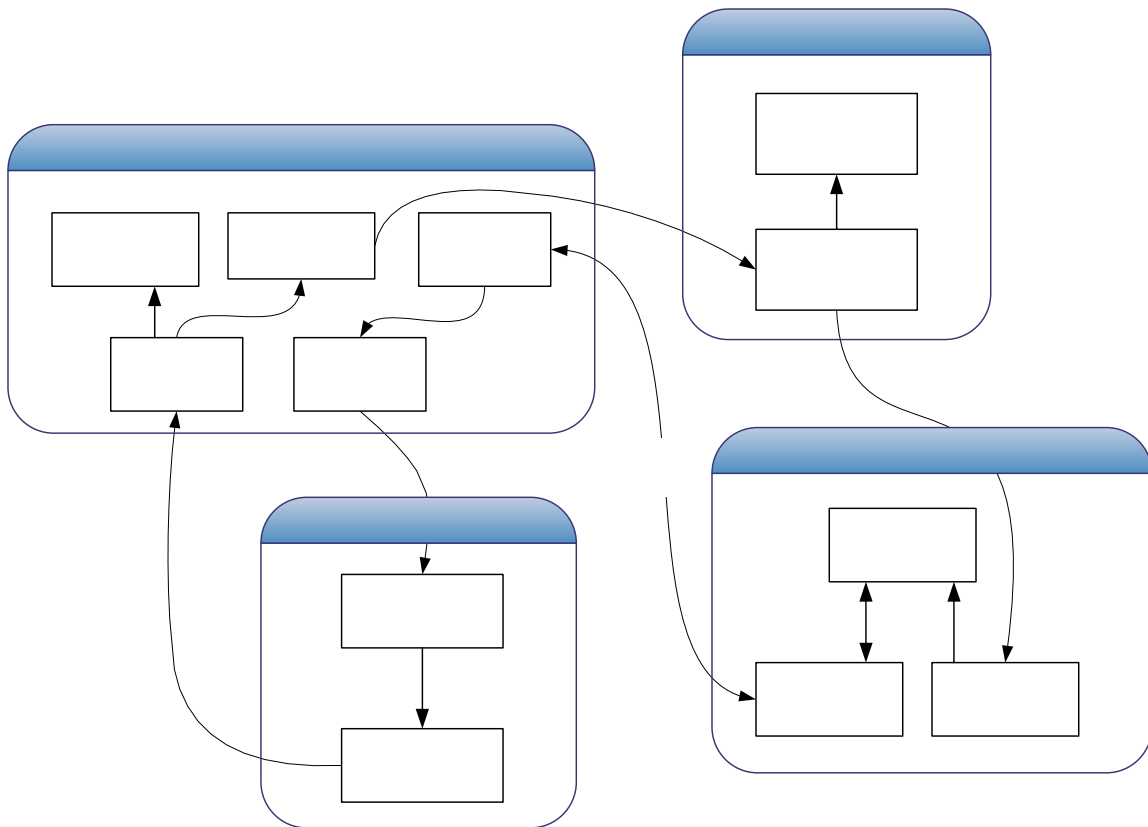


Figure 2: Software Components

The client and server communicate over the network using a simple ASCII protocol.

### 3.1. SCRAMNet DAQ Server

The server is an application written in both C and Java. The C part is responsible for interfacing with the SCRAMNet hardware and sampling data from it. This software component is known as the SCRAMNet interface. The Java part is the main application and controls the SCRAMNet Interface through a FIFO, called the control FIFO, using an ASCII protocol. The SCRAMNet interface send sampled data back to the main application in raw binary format using another FIFO, called the data FIFO.

This application is designed to run on Linux but most platform specific portions are contained in the C code.

#### 3.1.1. SCRAMNet Interface

This component performs two main functions:

- Listen for commands from the server over the control FIFO
- Send sampled data to the server over the data FIFO

##### 3.1.1.1. Control FIFO Protocol

These are the commands which the SCRAMNet interface support:

*sampleRate=X* – Set sample rate to X.

*numberOfChannels=X* – Set number of channels to X

*locations=X,Y,Z* – Set sample location to X,Y, and Z.

*triggerLocation=X* – Set trigger location to X.

*start* – Start acquisition loop. Sample data when trigger is high.

*Stop* – Stop acquisition loop.

*Exit* – Shutdown SCRAMNet interface.

*Rearm* – Rearm SCRAMNet interrupts. This is a workaround for a interrupt issue.

##### 3.1.1.2. Sampling and Data FIFO Protocol

The data fifo protocol simply send raw data read from SCRAMNet. The data is sent in individual samples and no formatting or delimiting is used. The data is 32 bit IEEE FP.

Sample format:

Timestamp

Sample 1

Sample 2

.....

And repeats for the next sample. There are two special types of data that are written to the data FIFO to indicate that a trigger has gone high or low. A data sample with a timestamp of 1 and data samples of all 0 indicate the trigger has gone high. A data sample with a timestamp of 2 and data samples of all 0 indicate the trigger has gone low. A data sample with a timestamp of 0 and data samples of all 0 indicate the acquisition has ended.

### 3.1.2. Data Listener

The data listener is responsible for collecting the data sent over the data FIFO and putting it in a queue. It is important to note that the data coming over the FIFO will be in the byte order of the local machine while Java always uses network byte order. So a conversion is done because Linux on x86 does not use network byte order.

Once the data is in the queue, it is available for use by the recording and streaming components.

#### 3.1.2.1. Recording

Once the data has been received it can be recorded to a file. This component takes data from the queue and writes it to a file in DADiSP format.

#### 3.1.2.2. Streaming

Once the data has been received it can be streamed to a RBNB server. This component takes data from the queue and streams it to a RBNB server using the RBNB API's.

### 3.1.3. Client Listener

The server is completely controlled by the client. This component listens on a port for connections from the client and processes commands from the client. The default port is 8012.

#### 3.1.3.1. Server Protocol

This is an ASCII protocol that controls all aspects of the server. These are the commands which it supports:

*Hello* – Must be first command to server

*Goodbye* – Must be last command to server

*Preview* – Get server ready to acquire data. SCRAMNet interface will be waiting for trigger. When triggered, data streaming will start.

*Start* – Get server ready to acquire data. SCRAMNet interface will be waiting for trigger. When triggered, data streaming and recording will start.

*Stop* – Stop data acquisition. Both data streaming and recording will stop.

*Set test name X* – Set current test name to X.

*Get test name* - Get current test name.

*Set channels X,Y,Z* – Set channels to sample. Where X, Y, and Z are channels to be sampled.

*Get channels* - Get channels to sample.

*Set sample rate X* – Set sample rate to X.

*Get sample rate* – Get sample rate.

*Set trigger source X* - Set trigger source to X, where X = {Table 1, Table 2, STS, Manual}

*Get test names* – Get test names of available data files.

*Get test: X* - Get data file associated with test name X.

### **3.2. SCRAMNet DAQ Client**

This is the UI to the server and is written in Java. It is build in two parts: The UI components and the server connection

#### **3.2.1. UI**

This is composed of Swing components handling configuration of the settings for data acquisition.

#### **3.2.2. Server Connection**

The talks to the server using the protocol defined in the Server Protocol (3.1.3.1). All configuration options are taken from the UI and sent to the server. When command buttons are pressed in the UI, these activate commands to be sent to the server to control its state of preview/start/stop.

## **4. Building**

Both applications use ant as a build system. The source files can be downloaded from: <http://nees.buffalo.edu/software/SCRAMNetDAQ/>

### **4.1. SCRAMNet DAQ Server**

This application was written for Linux and was tested on Red Hat Enterprise Linux 3.0. It is beyond the scope of this document to provide instructions for setting up SCRAMNet and its software. Please refer to documentation from Systran.

1. Download and install the J2SE JDK from <http://java.sun.com/j2se/1.4.2/download.html>. Set the environmental variable JAVA\_HOME to the directory you installed the JDK to. Add "\$JAVA\_HOME/bin" to your PATH.
2. Download and install Ant from <http://ant.apache.org/>. Set the environmental variable ANT\_HOME to the directory you installed Ant to. Add "\$ANT\_HOME/bin" to your PATH.
3. Download and install RBNB from <http://outlet.create.com/rbnb/>. Set the environmental variable RBNB\_HOME to the directory you installed RBNB to.
4. Install the SCRAMNet driver and software. Set the environmental variable SCRAMNET\_HOME to the directory you installed the SCRAMNet software to.
5. Unpack the SCRAMNet DAQ Server distribution and change to the unpacked directory.
6. Type "ant build" and everything should compile.

### **4.2. SCRAMNet DAQ Client**

This software is pure Java and should run on any system that supports Java. It has been tested on Windows and Linux.

1. Download and install the J2SE JDK from <http://java.sun.com/j2se/1.4.2/download.html>. Set the environmental variable

- JAVA\_HOME to the directory you installed the JDK to. Add “%JAVA\_HOME%/bin” to your PATH.
2. Download and install Ant from <http://ant.apache.org/>. Set the environmental variable ANT\_HOME to the directory you installed Ant to. Add “%ANT\_HOME%/bin” to your PATH.
  3. Unpack the SCRAMNet DAQ Client distribution and change to the unpacked directory.
  4. Type “ant build” and everything should compile.

## 5. Installation and Configuration

Note: There is no configuration necessary for the client.

### 5.1. SCRAMNet DAQ Server

There is a configuration file in *config/locations.txt* that configures the memory location that can be sampled. It maps a specific memory location and its attributes to a channel name and description that makes more sense. The file is tab delimited with one memory location per line. The format is:

Location<tab>	Source<tab>	Description<tab>	Type
---------------	-------------	------------------	------

The location is a word (32 bit) offset from the beginning of SCRAMNet memory. The source is a string describing what is generating this data. The description is a string which gives a human readable name to this location. Type is the data type of the location. Possible values are: int8, int16, int32, int64, float32, and float64.

## 6. Running

It is recommended that the server be started first and then the client. This is not necessary but it is a good idea to just keep the server running.

### 6.1. SCRAMNet DAQ Server

1. Go into the SCRAMNet DAQ Server directory and type “ant run” and the application will start and display debugging information.

### 6.2. SCRAMNet DAQ Client

1. Go into the SCRAMNet DAQ Client directory and type “ant run” and the application will start and display the UI.

See the SCRAMNet DAQ Users Manual for more information on using the client.

## 7. Authors

Jason P. Hanley  
 University at Buffalo  
 email: [jphanley@buffalo.edu](mailto:jphanley@buffalo.edu)  
 phone: (716) 645-2114 ext. 2450

## 8. Acknowledgements

This work is supported in part by the George E. Brown, Jr. Network for Earthquake Engineering Simulation (NEES) Program of the National Science Foundation under Award Numbers CMS-0086611 and CMS-0086612.

## 9. License

This software is released under the following license. This license is commonly known as the [MIT License](#).

Copyright (c) 2004 University at Buffalo

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

- The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.